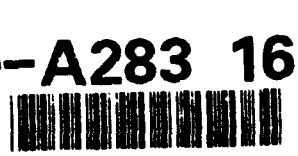


1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE
AN EXPERT NETWORK PROCESS CONTROL APPLICATION

5. FUNDING NUMBERS

6. AUTHOR(S)

DOUGLAS MICHEAL ROGERS

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Missouri

8. PERFORMING ORGANIZATION REPORT NUMBER

94-1060

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

DEPARTMENT OF THE AIR FORCE
AFIT/CI
2950 P STREET
WRIGHT-PATTERSON AFB OH 45433-7765

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for Public Release IAW 190-1
Distribution Unlimited
MICHEAL M. BRICKER, SMSgt, USAF
Cheif Administration

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

14. SUBJECT TERMS

15. NUMBER OF PAGES
114

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- **DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- **DOE** - See authorities.
- **NASA** - See Handbook NHB 2200.2.
- **NTIS** - Leave blank.

Block 12b. Distribution Code.

- **DOD** - Leave blank.
- **DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- **NASA** - Leave blank.
- **NTIS** - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

94-106

THESIS INFORMATION AND ABSTRACT

Title and author as they appear on title page:

AN EXPERT NETWORK PROCESS CONTROL APPLICATION

by

DOUGLAS MICHAEL ROGERS

Rank & branch of service: 1st Lieutenant - U.S. Air Force

Date: 1994

Number of pages: 114

Degree awarded: M.S. degree - Electrical Engineering

Institution: University of Missouri - Rolla

ABSTRACT

In this thesis, a computerized process control technique based on an artificial neural network (ANN) is presented. The process of concern is the wood dryer portion of an industrial retort used for the dehydration stage of wood waste pyrolysis in the making of charcoal. With the proposed control scheme, an ANN monitors the dryer system parameters, detects deviations from the norm, and applies corrective actions to counteract deviations should any occur. A standard backpropagation network structure is used to associate patterns of sensor readings with appropriate control responses that are based on process design specifications and operation heuristics. As heuristic programming is employed, the dryer controller is not unlike a typical knowledge base system, except that it is driven by an ANN rather than the more traditional rule base normally associated with expert system type applications.

A software simulated dryer unit is utilized to verify the functionality of a prototype network. Through simulation, it is shown that even when the ANN base is not familiar with a majority of the problem domain's input space, the expert network is capable, through generalization, of effective system level control for normal dryer operations as well as for cases where process faults are present.

BIBLIOGRAPHY

Juptner, Von J.H., Heat Energy and Fuels. New York: McGraw, 1908.

Johnson, A.J. and G.H. Auth, Fuels and Combustion Handbook. New York: McGraw-Hill, 1951.

Wenzl, Hermann F.J., The Chemical Technology of Wood. New York: Academic Press, 1970.

Miller, T.W., R.S. Sutton, and P.J. Werbos, Neural Networks for Control. Cambridge, MA: MIT Press, 1990.

Anderson K.L., Blankenship G.L., and Lebow, "A Rule Based Adaptive PID Controller," Proceedings of the 27th IEEE Conference on Decision and Control, I, 1 (1988), 564-569.

Ignizio, J.P., Introduction to Expert Systems: the Development and Implementation of Rule-Based Expert Systems. New York: McGraw-Hill, 1991.

Kosko, B., Neural Networks and Fuzzy Systems: A Dynamic Systems Approach to Machine Intelligence. Prentice-Hall, 1992.

Uhr, L.M., Multi-Computer Architectures for Artificial Intelligence. New York: John Wiley & Sons, 1987.

Badiru, A.B., Expert Systems Applications in Engineering and Manufacturing. Prentice-Hall, 1992.

Freeman, J.A., and D.M. Skapura, Neural Networks: Algorithims, Applications, and Programming Techniques. Addison-Wesley, 1992.

Antsaklis, P.J., "Neural Networks in Control Systems," IEEE Control Systems Magazine (April 1990): 3-5.

Chen, F.C., "Back-Propagation Neural Networks for Nonlinear Self-Tuning Adaptive Control," IEEE Control Systems Magazine (April 1990: 44-48.

Huang, S.C., and Y.F. Huang, "Learning Algorithms for Perceptrons Using Back-Propagation with Selective Updates," IEEE Control Systems Magazine (April 1990): 56-61.

Liu, H., T. Iberall, and G.A. Bekey, "Neural Network Architecture for Robot Hand Control," IEEE Control Systems Magazine (April, 1989): 38-42.

Passino, K.M., M.A. Sartori, and P.J. Antsaklis, "Neural Computing for Numeric-to-Symbolic Conversion in Control Systems," IEEE Control Systems Magazine (April, 1989): 44-52.

Chester, D., D. Lamb, and P. Dhurjati, "Rule-Based Computer Alarm Analysis in Chemical Process Plants," Proceedings of the Seventh Annual Micro-Delcon '84, Newark, (1984), 22-29.

THESIS INFORMATION AND ABSTRACT

Title and author as they appear on title page:

AN EXPERT NETWORK PROCESS CONTROL APPLICATION

by

DOUGLAS MICHAEL ROGERS

Rank & branch of service: 1st Lieutenant - U.S. Air Force

Date: 1994

Number of pages: 114

Degree awarded: M.S. degree - Electrical Engineering

Institution: University of Missouri - Rolla

ABSTRACT

In this thesis, a computerized process control technique based on an artificial neural network (ANN) is presented. The process of concern is the wood dryer portion of an industrial retort used for the dehydration stage of wood waste pyrolysis in the making of charcoal. With the proposed control scheme, an ANN monitors the dryer system parameters, detects deviations from the norm, and applies corrective actions to counteract deviations should any occur. A standard backpropagation network structure is used to associate patterns of sensor readings with appropriate control responses that are based on process design specifications and operation heuristics. As heuristic programming is employed, the dryer controller is not unlike a typical knowledge base system, except that it is driven by an ANN rather than the more traditional rule base normally associated with expert system type applications.

A software simulated dryer unit is utilized to verify the functionality of a prototype network. Through simulation, it is shown that even when the ANN base is not familiar with a majority of the problem domain's input space, the expert network is capable, through generalization, of effective system level control for normal dryer operations as well as for cases where process faults are present.

BIBLIOGRAPHY

Juptner, Von J.H., Heat Energy and Fuels. New York: McGraw, 1908.

Johnson, A.J. and G.H. Auth, Fuels and Combustion Handbook. New York: McGraw-Hill, 1951.

Wenzl, Hermann F.J., The Chemical Technology of Wood. New York: Academic Press, 1970.

Miller, T.W., R.S. Sutton, and P.J. Werbos, Neural Networks for Control. Cambridge, MA: MIT Press, 1990.

Anderson K.L., Blankenship G.L., and Lebow, "A Rule Based Adaptive PID Controller," Proceedings of the 27th IEEE Conference on Decision and Control, I, 1 (1988), 564-569.

Ignizio, J.P., Introduction to Expert Systems: the Development and Implementation of Rule-Based Expert Systems. New York: McGraw-Hill, 1991.

Kosko, B., Neural Networks and Fuzzy Systems: A Dynamic Systems Approach to Machine Intelligence. Prentice-Hall, 1992.

Uhr, L.M., Multi-Computer Architectures for Artificial Intelligence. New York: John Wiley & Sons, 1987.

Badiru, A.B., Expert Systems Applications in Engineering and Manufacturing. Prentice-Hall, 1992.

Freeman, J.A., and D.M. Skapura, Neural Networks: Algorithims, Applications, and Programming Techniques. Addison-Wesley, 1992.

Antsaklis, P.J., "Neural Networks in Control Systems," IEEE Control Systems Magazine (April 1990): 3-5.

Chen, F.C., "Back-Propagation Neural Networks for Nonlinear Self-Tuning Adaptive Control," IEEE Control Systems Magazine (April 1990: 44-48.

Huang, S.C., and Y.F. Huang, "Learning Algorithms for Perceptrons Using Back-Propagation with Selective Updates," IEEE Control Systems Magazine (April 1990): 56-61.

Liu, H., T. Iberall, and G.A. Bekey, "Neural Network Architecture for Robot Hand Control," IEEE Control Systems Magazine (April, 1989): 38-42.

Passino, K.M., M.A. Sartori, and P.J. Antsaklis, "Neural Computing for Numeric-to-Symbolic Conversion in Control Systems," IEEE Control Systems Magazine (April, 1989): 44-52.

Chester, D., D. Lamb, and P. Dhurjati, "Rule-Based Computer Alarm Analysis in Chemical Process Plants," Proceedings of the Seventh Annual Micro-Delcon '84, Newark, (1984), 22-29.

UNIVERSITY OF MISSOURI-ROLLA

School of Engineering

Department of Electrical Engineering

[illegible]

12 May 94

To Whom It May Concern.

I have served as a graduate advisor to Douglas Rogers while he was completing his MS degree in Electrical Engineering at the University of Missouri-Rolla. I also served as his instructor for two of his graduate courses-EE313 and EE412.

He did excellent work on all of his classes. He was the top student in my EE313 class on Microprocessor System Design. I bring each student in for several oral exams over the laboratory projects that have been assigned in EE313. He impressed me with how well he understood the techniques and basic concepts. He is well organized. At times I would use his work on the assigned projects and homework as the answer key while I was grading the other students work. He was able to complete the seven hardware and software experiments with a minimum amount of help and supervision. One experiment was done on the control of stepper motors.

He was also the top student out of the twelve graduate students in my EE412 Advanced Logic Design Course and Switching Theory Course. In addition to the classical material presented on the foundations of logic design-Boolean Algebra, Karnaugh Maps and minimization techniques, Mr. Rogers completed two special projects on the characteristics of electronic gates and flipflops. He was also introduced to some of the programming techniques used on Programmable Logic Devices (PLD's).

I was co-advisor along with Dr.Dagli for his MS thesis: " An Expert Network Process Control Application". In this thesis a control scheme was designed for the modernization of the wood dryer portion of a charcoal plant. Techniques from two new fields, Expert Systems and Neural Networks, were used as the basis of this control system. The system design concepts learned in this design project could be applied to a great variety of projects in the future.

He did fine work in the design and write up of his thesis work. He gave one of the best oral presentation on his thesis that I have attended in a number of years.

He wrote and presented a paper "Retort Optimization Through Neuro-Control Approach" at the Conference on Computer Integrated Manufacturing in the Process Industries held at East Brunswick, New Jersey on April 26th, 1994. This Conference was hosted by Rutgers University.

He has been dedicated to his graduate program. I would rate him as being in the top 3% of our graduate students who have obtained degrees in the past eight years.

It has been a pleasure to work with him. Please call me if any additional information is needed.

Sincerely

Darrow F. Dawson
Professor and Graduate Area Coordinator

ADVISOR/INSTRUCTOR COMMENTS

I was fortunate to co-advise Douglas M. Rogers towards his MS degree in Electrical Engineering. The research he has done on intelligent process control is an original piece of work which has evolved from real life problems. In this work Doug was able to integrate new techniques that are introduced to him in the program; namely; expert systems and artificial neural networks effectively for the solution of a process control problem. The work done will be a source of inspiration to a lot of companies as it was to the company that we worked with closely during the thesis work.

The originality of the work is also endorsed by the acceptance of a paper titled "Retort Optimization through Neuro-Control Approach" co-authored by Doug and myself for the CIMPRO'94 Conference on Computer Integrated Manufacturing in the Process Industries. Doug presented the paper on April 25-26, 1994 in East Brunswick, New Jersey. A journal paper that summarizes the research is currently under review for the Journal of Integrated Computer-Aided Engineering.

Doug is a goal oriented, hard working, dependable individual with a research oriented mind. He is a team player with a nice personality. I enjoyed having him as a student and wanted to keep him for further degrees but I was not able to compete with his desire to fly.

Dr Cihan Dagli

703-313-0514

AN EXPERT NETWORK PROCESS CONTROL APPLICATION

by

DOUGLAS MICHAEL ROGERS, 1968 -

A THESIS

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI - ROLLA

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

1994

Approved by

Darrow F. Dawson, Co-Advisor

Cihan H. Dagli, Co-Advisor

Paul D. Stigall

AN EXPERT NETWORK PROCESS CONTROL APPLICATION

by

DOUGLAS MICHAEL ROGERS, 1968 -

A THESIS

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI - ROLLA

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

1994

Approved by

Darrow F. Dawson, Co-Advisor

Cihan H. Dagli, Co-Advisor

Paul D. Stigall

ABSTRACT

In this thesis, a computerized process control technique based on an artificial neural network (ANN) is presented. The process of concern is the wood dryer portion of an industrial retort used for the dehydration stage of wood waste pyrolysis in the making of charcoal. With the proposed control scheme, an ANN monitors the dryer system parameters, detects deviations from the norm, and applies corrective actions to counteract deviations should any occur. A standard backpropagation network structure is used to associate patterns of sensor readings with appropriate control responses that are based on process design specifications and operation heuristics. As heuristic programming is employed, the dryer controller is not unlike a typical knowledge base system, except that it is driven by an ANN rather than the more traditional rule base normally associated with expert system type applications.

A software simulated dryer unit is utilized to verify the functionality of a prototype network. Through simulation, it is shown that even when the ANN base is not familiar with a majority of the problem domain's input space, the expert network is capable, through generalization, of effective system level control for normal dryer operations as well as for cases where process faults are present.

ACKNOWLEDGEMENTS

I would like to thank Dr. Cihan H. Dagli for his guidance and for allowing me to take part in the project that generated the research presented in this thesis. I would also like to thank my department advisor, Dr. Darrow F. Dawson, whose time and advice made my transition back to academia so much easier. Next I would like to thank Dr. Paul D. Stigall for reviewing my work and serving as a committee member. Finally, I would like to thank my wife Peggy for her much needed support throughout our stay at UMR.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

I. INTRODUCTION

A. BACKGROUND AND INTENTIONS

The use of artificial intelligence (AI) techniques in industrial process control applications has become quite widespread. From expert systems to neural networks, AI technologies are "all the rage," the latest revelation in industry's never ending quest for newer and better control strategies that enhance productivity.

In this case, a charcoal manufacturer desires a computerized process monitoring system to aide in the operation of a retort currently under construction. At plant locations that already have working retorts, the company relies solely on human operators to make the various system level adjustments necessary to keep the retort subprocesses functioning properly. There are no standard operating procedures and very few written documents available to support or guide the operators in their decision making. At any given time, therefore, productivity is a function of the skill and experience of the particular operator on duty.

The resulting nonuniformity in control practices has always been acceptable because production efficiency has not been a major consideration in retort operations. This is primarily due to the fact that the sole process raw material, wood waste, is extremely cheap and readily available. As long as the retorts produce char (the end

product of the process), there has been little concern about the efficiency with which they do it.

Lately, however, the growing use of wood waste by other manufactures (e.g. paper mills and competing charcoal producers) has resulted in greater demand for the bountiful raw material. With this ever increasing competition for wood waste, the company realizes it will soon be economically advantageous to pay more attention to the efficiency of retort operations. Therefore, the company has decided to overhaul its process control procedures and implement the refurbished program when the retort now under development goes on line. The major intended change involves the addition of computers to aide in operator decision making.

Computerized control procedures provide standardization and, as long as sound control methods are encoded, predictably efficient operations; this is what the company hopes to achieve with its modernization effort. Initially, a traditional expert system approach was considered the best choice to satisfy the company's aims. Given the particulars of this problem domain, such as the complexity of the retort system and lack of existing standardized control procedures, it was soon evident that a typical expert system solution would require an extensive knowledge base. To avoid this, a neural network was selected to drive the expert system. The resulting "expert network," as it is termed, is a hybrid combination of these two technologies. The goal of the

research contained in this thesis, therefore, is to demonstrate, through developmental analysis and software implementation, the potential this somewhat unconventional AI approach has for solving the problem at hand.

B. PROBLEM DOMAIN

1. Process Basics. A retort performs the partial thermal decomposition of wood waste. Stated more simply, a retort burns sawdust in a controlled manner to specified carbon, ash, moisture, and volatile levels. Various by-products such as gases and tars are driven off leaving a clean burning residue comprised mostly of carbon [1]. This end product (called char) is used in a separate process (briquetting) to make the familiar barbecuing type charcoal.

The major functional units of a retort, shown in Figure 1, are a wood dryer that drives off most of the sawdust's moisture and a multiple hearth furnace that carbonizes the dried sawdust. There are also numerous conveying systems and storage bins, but the control issues affecting the quality of the char produced stem from the operation of the wood dryer and furnace systems.

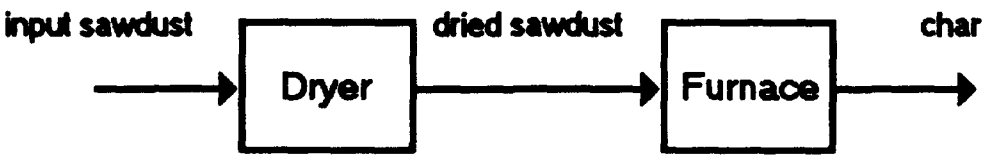


Figure 1. General Retort Process

While several parameters (e.g. wood type, particle size/shape, furnace temperatures, etc..) affect the caliber of char produced by pyrolysis, moisture is the most significant of the process variables. Typical moisture ranges are 45-55% for input sawdust [2], 8-10% for the dryer output sawdust [3], and 12-15% for char leaving the system [1]. The dryer acts as a moisture filter for the furnace which can tolerate only a narrow input moisture range if it is to adequately char the sawdust. Regardless of the range of moisture entering the system (which can be anywhere from practically dry to extremely wet depending on environmental conditions), therefore, the dryer must be able to dehydrate the input sawdust to the specified range. This is accomplished by exposing the sawdust to a temperature slightly above the boiling point of water. It is then left to the furnace to complete the process by carbonizing the dried sawdust using much higher temperatures. These temperatures typically range from 300-400°C resulting in a 37-51% char yield that is roughly 80% carbon, 1-3% ash, and 12-15% volatile matter [3].

2. Scope of Solution. The retort system briefly described above, while simple in purpose, is actually quite complex in terms of the number and type of subprocesses required for actual implementation. Particle sizers, wood waste transportation devices, dryer components, furnace components, and a host of other mechanisms all must be adequately controlled for the system to function properly.

To aide in operations, the company desires a computerized monitoring system capable of high level decision analysis. Such a sophisticated program results only after the steps of a well-ordered design process are successfully accomplished. The approach taken for this project is one of divide and conquer. That is, the proposed solution methodology described in this thesis is applied to only a portion of the retort process so that verification procedures on a smaller, more manageable domain are easier to accomplish thus paving the way for future applications at higher, more general, levels.

The dryer section of the retort was selected as the test module because it requires the most operator attention in terms of control adjustments. As the actual dryer to monitor is not yet operational, a hypothetical dryer unit is modeled in software. Although small scale simulations often lack the complexities present in real-world processes, they are a good starting point for testing because of the flexibility the designer has in determining what scenarios to examine and how to go about it. Since, as previously mentioned, there is a lack of technical data and procedural information regarding system operations, the makeup of the dryer simulator (to be discussed in greater detail shortly) is based on a number of sources, some more technically accurate than others. Hopefully, what the model lacks in real-world accuracy the solution architecture makes up for in generality so that its applicability in other problem

domains is not limited by any assumptions on which the current process simulation is based. The scope of the research presented in this thesis, therefore, is solution verification through simulation. The results should enable the next phase of the project, the application of the expert network design to an actual process.

C. SECTION ORGANIZATION

In Section II, the basic theory behind neural networks and expert systems is discussed. Also, the advantages and disadvantages associated with each of these AI technologies in terms of control applications are presented. S ction III describes the development of a software simulated process environment and a prototype expert network used to monitor and control this simulated process. Section IV reviews the efficiency and effectiveness of the prototype expert network in its operation. Finally, Section V provides a summary of results, a few system enhancement considerations, and some recommendations regarding future research for the project on which this thesis is based.

II. ARTIFICIAL INTELLIGENCE AND CONTROL

There are numerous computer aided process control schemes currently available to manufacturers; each has its relative strengths and weaknesses in terms of cost, accuracy, speed, complexity, and so on. Most techniques are founded on the use of mathematical models to achieve optimal control solutions [4]. Arriving rather recently on the process control scene, expert systems and neural networks are AI based technologies now commonly used as stand alone controllers as well as the implementation mechanism for more traditional control algorithms[4][5]. Expert systems are objective driven procedures relying on symbolic processing rather than mathematical relationships to reason their way to solutions. Neural networks, on the other hand, developed from conceptualizations of how the human brain functions and are noted for their associative learning capabilities.

A. CONTROL THEORY BASICS

The foundation of control theory is mathematical analysis. The general controller design procedure involves identification of the system to control, the development of a mathematical definition of this system, and the determination of adequate control solutions based on the system variable relationships enumerated by this mathematical definition. Since, ideally, all aspects of the system's domain are quantified or accounted for, optimal

control solutions are derived through the application of an algorithm to the mathematical model [6]. The true value of an algorithmic solution depends, of course, on how accurately the model represents the actual system. Most device and small system level controllers utilize algorithmic methods for optimal control since the mathematical models at these lower levels are usually of manageable size.

B. EXPERT SYSTEMS

Expert systems are goal oriented procedures that attempt to solve problems via a step by step reasoning process defined in what's termed its inference engine. The inference engine makes decisions based on user inputs and pre-encoded information contained in a knowledge base (Figure 2).

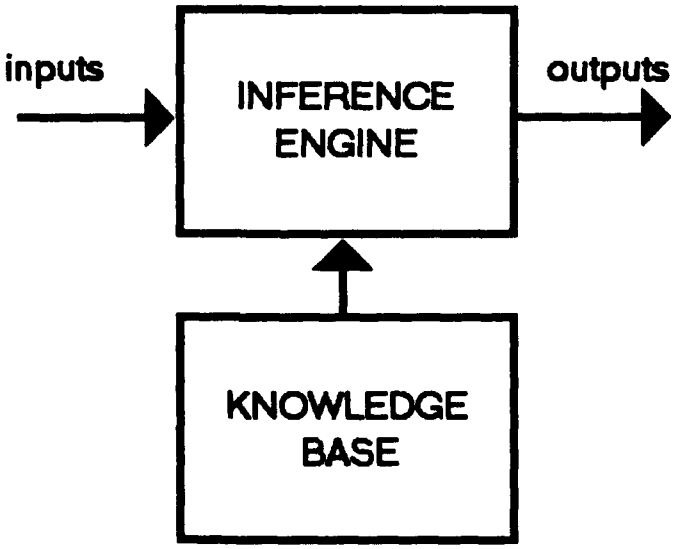


Figure 2. Generic Expert System

With an expert system, information is normally represented symbolically and processed using propositional rules that take the logical implication form: IF PREMISE THEN CONCLUSION [7]. Reasoning, therefore, is accomplished utilizing a predicate calculus where the truth values of rule premises trigger corresponding conclusions that eventually lead to an overall system state deduction.

The method by which the propositional rules are evaluated is called the inferencing strategy. This is how an expert system logically deduces a conclusion from the information it is given and the rules contained in its knowledge base. The two most common processes used are forward chaining and backward chaining. The difference between these methods is subtle, but certain applications require the use of one approach over the other. Figure 3 shows an example where both chaining types are used to process a single rule. With forward chaining, the truth values of the three premises A, B, and C are evaluated first, and the status of conclusion D is then determined. With backward chaining, conclusion D is initially sought as a goal, so premises A, B, and C are evaluated in an attempt to conclude the status of D. Obviously, for process control applications, forward chaining is generally preferred as this method more closely matches the input(s) affecting output(s) type relationship present in most systems.

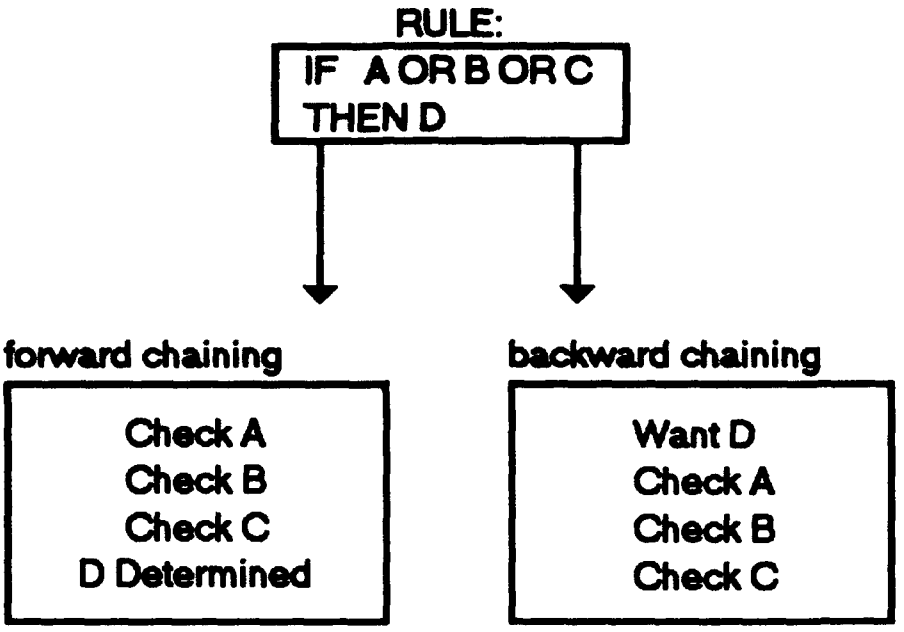


Figure 3. Inferencing Techniques

The knowledge base is usually made up of heuristics, experience or judgment based rules of thumb not necessarily supported by tangible evidence [6]. Unlike the algorithmic approach previously discussed, applying a heuristic based procedure to a control problem does not guarantee process optimization [8]. That is, depending on the quality of the heuristics contained in the knowledge base and the inferencing strategy employed, the expert system may or may not come up with the mathematically optimal solution (or even correct solution) that, say, a state-variable representation and analysis of the same system would. Certain classes of problems (termed combinatorial explosive), however, are too large to develop accurate

mathematical models for (given realistic time and effort constraints). Large scheduling tasks, the classic traveling salesman scenario, and the game of chess are examples of problems where exhaustive solution searches are impractical. Expert systems, therefore, are often used with the hope of heuristically solving these types of problem.

When used as active controllers, expert systems are required to perform their reasoning functions in real time [5]. Processing information symbolically in software, therefore, is a major disadvantage because it precludes the direct use of large scale integrated circuits [7]. As such, expert controllers are generally slower than comparable algorithmic solutions that can often be implemented directly in hardware. If heuristic programming is the only practical solution approach, however, this shortcoming is irrelevant. Expert system control, therefore, is most appropriate for those applications controlled by human experts where the problem at hand is too big or complex to completely specify mathematically and, hence, solve algorithmically [9].

Such is the case with the industrial retort. For a general system level analyzer, it is simply not feasible (in terms of time, cost, and effort) to derive the necessary relationships between the numerous process variables in an attempt to apply an algorithmic control solution. Therefore, it was decided early on to use an AI approach in the development of the desired decision support system.

C. NEURAL NETWORKS

1. General Overview. Neural networks evolved from research that sought to mimic the capability of the human brain to learn. Most ANN architectures are composed of an interconnection of nodes which are not unlike but, at the same time, not intended to be exact models of brain cells and the neural links between them. Since computer clock periods run much faster than even the quickest stimulus-to-reaction network found in the brain (nano versus milli second cycle times), the potential for computer applications based on neural models is obvious [8].

Essentially composed of memory type cells, most neural networks "learn" through association, and this associative learning base is what gives the neural network its tremendous pattern recognition abilities [10]. Basically, an ANN relates a given input or set of inputs with a particular output or set of outputs. There are, however, many different ANN architectures; each of which has a particular nodal structure and method for input-output pattern representation. There are, however, only two basic methodologies for teaching or training an ANN: supervised and unsupervised learning.

An ANN that utilizes a supervised training scheme is taught by presenting patterns of inputs to the network and adjusting the various network parameters until the desired network output is achieved. This is accomplished for a representative sample of all input-output combinations.

Hopefully, after a certain number of examples, the network will be able to generalize correct output response for all inputs, even for those it has not been exposed to during training.

Unsupervised learning involves the classification of input patterns into clusters or groups containing similar characteristics [7]. There are no predetermined input-output relationships; only network inputs are known and the output produced represents the group or class an input belongs to with respect to the other inputs with which the ANN is familiar. Obviously, this method is used where particular ANN outputs are not required or set in advance.

Both supervised and unsupervised learning are implemented in many ways. Different ANN paradigms (e.g. backpropagation, bidirectional associative memory, adaptive resonance theory, adaptive critic, etc..) have distinct ways of computing the various values associated with the individual nodes, nodal connection links, and any number of other elements which may be part of a given network structure. With many neural network "flavors," proper paradigm selection depends on the specifics of the intended application.

2. Backpropagation. Neural networks based on the backpropagation learning scheme are characterized by layers of interconnected, nonlinear elements (neurons) that accept one or more inputs and generate an output.

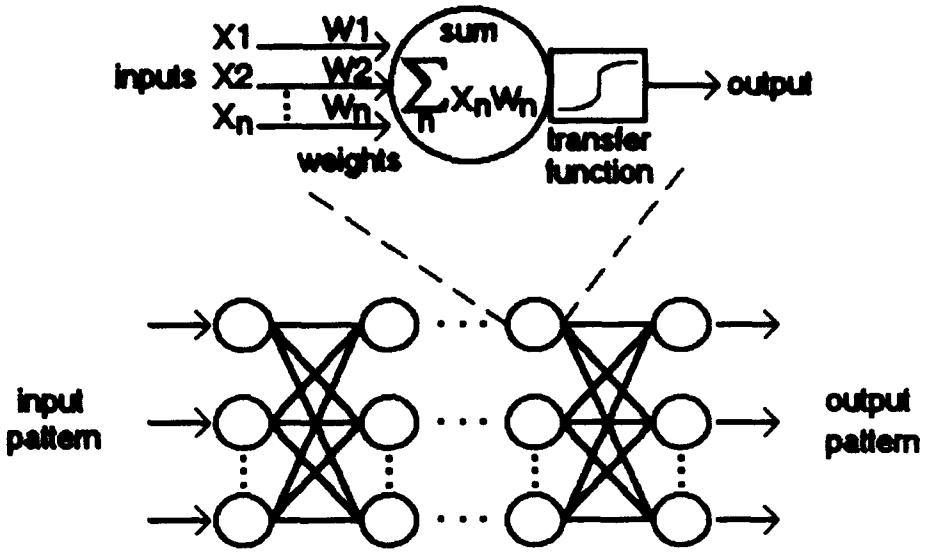


Figure 4 Standard Backpropagation Network

As seen in Figure 4, each neural input has a corresponding weight associated with it. Neurons process information by summing the weighted input signals and using the result to produce an activation signal based on a given transfer function. The desired output representation dictates the type of transfer function used. For binary outputs, the sigmoid (logistic) function is typically used so that outputs range between 0 and 1 [11]. Since the outputs can never actually reach 0 or 1, binary high and low must be redefined (e.g. 0.1 for 0 and 0.9 for 1). The output of an individual neuron of Figure 4, therefore, is given by the following equation:

$$\text{output} = (1+e^{-\text{sum}})^{-1}$$

In this way, the outputs of each nodal layer are the inputs to the next layer until the final layer produces the network output.

Typically, first layer neurons are characterized by single inputs and unity weight/transfer function values. Network inputs, therefore, pass through this layer unchanged. The number of input and output layer neurons in a particular network varies with the representation desired for a given application. There is no standard procedure for determining the correct number of neurons in the middle, or hidden layers; a network with too few hidden nodes, however, will not converge to an error minima (train adequately) [10]. If a network is not learning properly, therefore, arbitrarily adjusting the number of neurons in the hidden layer(s) is one way to possibly improve performance.

When using a backpropagation based ANN, the goal is to obtain a particular network output for a given input. To accomplish this, backpropagation networks utilizes the generalized delta rule (GDR), a supervised learning procedure for training the ANN to make desired input-output associations. In short, this procedure involves manipulation of the neuron input weights until the desired network output is achieved for all input patterns presented to the network during training.

Consider the three layer network shown in Figure 5 (neuron sums and outputs are computed as shown in Figure 4 with a sigmoid transfer function used).

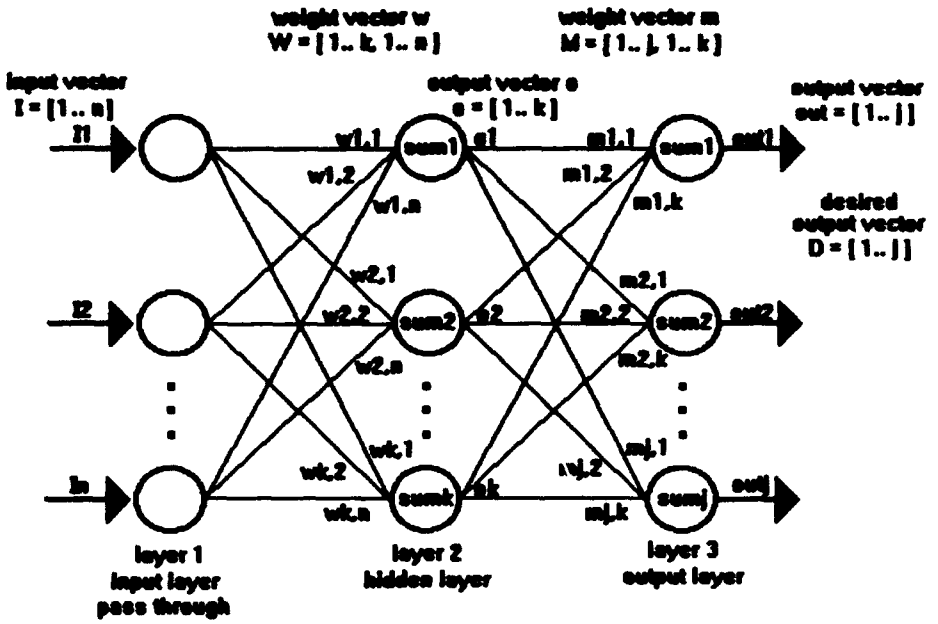


Figure 5. Three Layer Backpropagation Network

To begin training with the GDR, input patterns, or vectors (I) are presented to a network initialized with randomized weight vectors (W and M), and an error value between the network output (out) and the desired output (D) is computed for each of the j output neurons.

$$\text{output error}_j = (\text{out}_j - D_j)\frac{e^{-\text{sum}_j}}{(1 - e^{-\text{sum}_j})^2}$$

Next, the error value is calculated for each of the k hidden neurons.

$$\text{hidden error}_k = \frac{e^{-\text{sum}_k}}{(1 - e^{-\text{sum}_k})^2}\sum_j \text{output error}_j M_{j,k}$$

The weights vectors are then incrementally adjusted in the negative gradient direction of the hyperspace defined by the error-weight relationships until the proper output is generated. For **W** and **M** this is accomplished with

$$W_{k,n}(t+1) = W_{k,n}(t) + \eta \text{ hidden error}_k \; I_n$$

and

$$M_{j,k}(t+1) = M_{j,k}(t) + \eta \text{ output error}_j \; o_k$$

respectively, where η is a learning parameter governing the amount of weight change per iteration (usually a value between 0.05 and 0.25) [10]. Tolerance is a measure of the overall output error value. This is given by

$$\text{tolerance} = \frac{1}{2} \sum_j \text{output error}_j^2$$

When the tolerance is low enough for all input vectors applied to the network, training is complete.

After applying the GDR procedure, input-output relationships are recalled merely by applying an input vector to the system; this is accomplished by simple vector multiplication.

$$\text{out} = [IW]M$$

If the ANN is trained to a low enough tolerance, all of the training patterns presented to the network will be

recognized and the corresponding desired output will appear in **out**.

For input vectors not used in training, what the network produces as an output is not so cut and dry. Without applying the entire input space to the ANN, there is no way of knowing how the network will respond to unfamiliar inputs [7]. Hopefully, the training vectors represent a sufficient cross-section of the input-output relational domain so the network is able generalize its responses and achieve a high degree of accuracy for all input-output associations. If not, the network can be trained to a lower tolerance or retrained entirely using different parameters (e.g. learning rate, initial weights, training vector, etc.).

In terms of process control applications, the advantages offered by neural networks in general and backpropagation in particular are quite substantial. Basically, the overall network structure lends itself to parallel processing and fast hardware implementation for real-time applications, the GDR procedure of backpropagation can be recursively implemented in parallel hardware, and any function can be represented to virtually any accuracy provided a large enough network is used [4].

Backpropagation does, however, have its shortcomings. With its gradient descent method of "learning" input-output associations, large backpropagation networks tend to take a long time to train [12], and training is by no means an

exact science. Backpropagation networks, for example, can forget old knowledge when improperly trained with new information, and considerations such as learning rate, testing data, the number of hidden neurons, and a host of other parameters all affect how successfully a network trains [10]. Also, the GDR learning algorithm tends to converge at local rather than global error-weight minima which makes for less accurate networks [13]. Many of the newer, more sophisticated ANN architectures have eliminated a number of these difficulties; but, as is always the case, application specifics inevitably dictate paradigm selection as all ANN techniques have their strengths and weaknesses. Such is the case with the prototype expert network development discussed in the next section.

III. EXPERT NETWORK DEVELOPMENT

A. ADVANTAGES OF SOLUTION APPROACH

As mentioned, a retort is a relatively complicated system. With hundreds of individual sensor measurements, considering all possible combinations in an attempt to formulate the interrelationships between them would be a formidable task to say the least. Since the foundation of an expert system, however, is search space reduction through the use of heuristics, and a neural network is very adept at pattern recognition, it was decided to develop an expert network that took advantage of these characteristics in its solution approach. First, the multi-variable search space of the process is reduced through the use of operation heuristics to identify dominant system variables. Next, the ANN learns to associate proper control responses with the system states defined by these variables.

Basically keying on sensor data to determine process states, the ANN employed by the expert network must accept analog inputs, have a heteroassociative memory, be good at generalization in the face of unfamiliar inputs, and be able to store and quickly process large amounts of data. These are the characteristics of the backpropagation architecture which is selected for the ANN implementation. While not in vogue among neural network architectures, backpropagation has been used successfully in numerous applications and, perhaps more than any other architecture, has been

thoroughly researched with regards to its capabilities and limitations; it has, in fact, become a standard benchmark with which other architectures are compared. Also, because of its basic multi-layer feed forward structure, the conversion to other paradigms, should it become necessary and/or beneficial to do so, is not difficult.

In terms of expert control, the parallel processing nature of a neural network is an extremely attractive feature since symbolic processing can limit the usefulness of expert systems for real-time control applications because of speed considerations. With the expert network, therefore, replacement of the relatively slow rule base with a potentially faster ANN base allows for the possibility of a significant speed advantage given the ever growing use and development of neuro-hardware.

Instead of using propositional rules as do traditional expert controllers, the ANN base of the expert network relies on input-output (sensor-control) relationships developed from operation heuristics and imparted to the network via training sessions. In this way, changes and updates to the expert network simply require the addition, deletion, or redefinition of an input-output relationship and the retraining of the network. While this may not be as straightforward as it sounds, the concept is certainly simpler than the task of updating a linearly programmed knowledge base where the intricate interrelationships

between rules and goals must be accounted for every time changes are made.

When developing an expert system, the entire input space must be considered so the knowledge base knows what to do for at least all situations where uncertainty factors are not involved. Even with heuristic programming, high order applications, those with many states and numerous variables, may require an extensive knowledge base. An adequately trained backpropagation ANN base, however, is able to generalize responses to unfamiliar stimuli based on its learned input-output associations. In many cases, such generalization is achievable even which the network is trained over only a small portion of the system input space [14]. This is a key feature of an ANN, especially in those cases where it is impossible to cover all input-output relationships due to the size of the system, a lack of knowledge, or both.

For control applications, therefore, expert networks have many potential advantages over conventional knowledge base systems. Almost all of these advantages (speed, maintainability, and generalization ability) stem from the capabilities of the backpropagation ANN. In fact, the only portion of traditional expert system theory involved in this project is heuristic programming. The proposed solution, therefore, is perhaps more of a heuristic network than an expert network.

B. PROTOTYPE SIMULATION.

The prototype for the retort monitor is a small scale expert network designed to oversee the running of a software simulated retort dryer unit called DRYER-SIM. In addition to monitoring dryer operations, the expert network is tasked with using its knowledge of the state of the dryer to actively control certain subprocesses. This serves two purposes. First, it represent operator implementation of control recommendations. This frees the user to concentrate on testing scenarios, eliminating the need for someone to play operator and push simulated buttons every time the network determines the need for system adjustments. Second, for the research purposes of this thesis, it makes for a more interesting problem and further demonstrates the abilities of the expert network.

1. DRYER-SIM Model. Obviously, the development of a model for simulation eliminates the need for control heuristics since the DRYER-SIM operation specifications are readily available, defined in the model. The expert network developed for the simulator, therefore, can be viewed as possessing a complete set of accurate dryer operation heuristics, something that will undoubtedly take a long time to compile for the full-scale implementation.

While much effort has been made to keep DRYER-SIM realistic, the main focus of this simulation is to give the prototype expert network a reasonable non-linear system to

respond to rather than produce an extremely accurate model of the sawdust drying process.

a. Operation Overview. DRYER-SIM is based on the hypothetical wood-waste dryer shown in Figure 6 below.

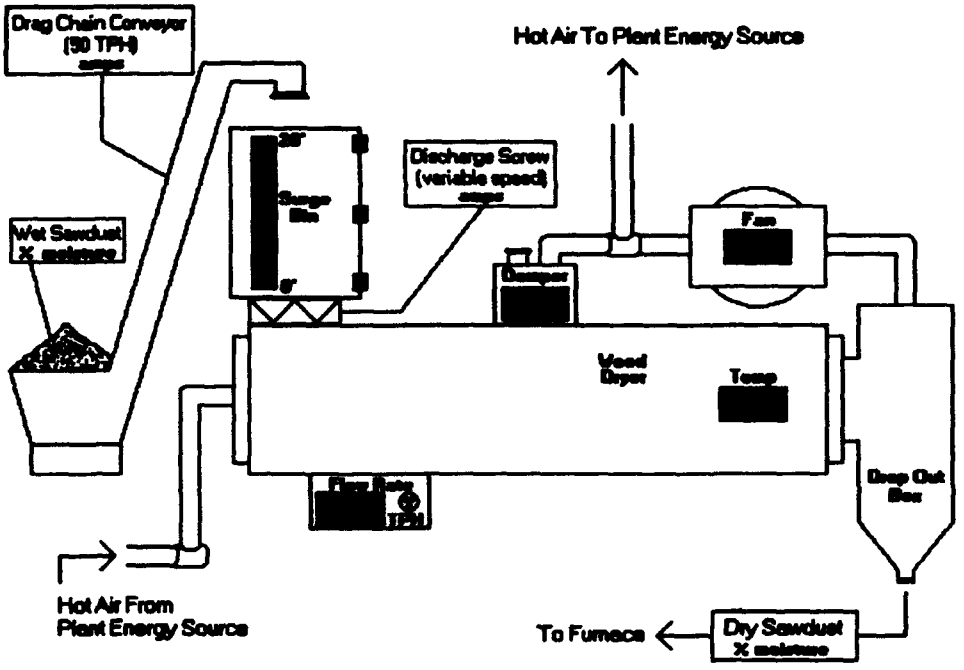


Figure 6. DRYER-SIM Environment

Pre-sized wet sawdust enters the system via the drag chain conveyer that operates in an on/off manner to keep the surge bin filled about the 10 foot level. There are three level sensors placed at the 5, 10, and 15 feet marks to keep track of the amount of sawdust in the bin. Sawdust from the surge bin is fed to the dryer through the discharge screw at rate dependent on the amount of moisture in the sawdust. The dryer temperature range is set to 220±5°F. High moisture

sawdust takes longer to dry at 220°F, so the dryer flow rate is slower than when sawdust lower in moisture is being processed. The dryer temperature is maintained by a fan that draws heat through the dryer from a plant-wide energy source. Higher fan speeds cause more hot air to flow through the dryer and, hence, higher temperatures; lower fan speeds have the opposite effect. Sawdust leaves the dryer through the drop out box at an 8-10% moisture level assuming the drying is operating normally. Abnormal operations, of course, have various affects on the output sawdust moisture range.

b. Assumptions. The DRYER-SIM program is written in an object oriented Production Rule Language supported by the LEVEL5 OBJECT expert system developmental software (see user's manual in APPENDIX A for system requirements and instructions). As with most software models, a number of simplifying assumptions are made in order to keep the program of manageable size.

First of all, so the evaporation point of water is a constant 212°F, standard day atmospheric conditions (at sea level) are assumed. Also, although the sawdust moisture content is variable, particle geometry is uniform at 0.5 in^3 and average sawdust density is 10 lb/ft^3 [2]. Ambient temperature and relative humidity, moreover, have no effect on DRYER-SIM operations. Finally, all DRYER-SIM systems not provided with a failure option always operate ideally. Less

general, system particular assumptions are listed as the individual system specifications are described.

c. Specifications. The mathematical relationships that define the workings of the DRYER-SIM are relatively straight forward and are presented subsystem by subsystem.

The height of the surge bin is described by the first-order nonlinear differential equation

$$dh = k\left(\frac{dI'}{dt} - \frac{dO}{dt}\right)dt$$

where h is the height of the sawdust in the bin, k is a volume constant equal to 0.003, dO/dt is the sawdust flow rate out of the bin (dryer flow rate), and dI'/dt is the input sawdust flow rate given by

$$\frac{dI'}{dt} = 10000\,\frac{ft^3}{hr}\,I(t),\quad I(t) = \begin{cases} 0 \text{ if drag chain off} \\ 1 \text{ if drag chain on} \end{cases}$$

To keep a certain amount of sawdust in the bin, therefore, the constant speed drag chain must be switched on and off appropriately to offset the variable outflow of sawdust given by dO/dt. This outflow, or dryer flow rate is a function of the sawdust moisture level and is depicted in Figure 7. At its maximum processing speed (50 TPH), the dryer flow rate is exactly matched by an operating drag chain. At higher sawdust moisture levels, the input sawdust rate of the drag chain is greater than the dryer flow rate. The non-zero slope of the function in Figure 7 is taken from [2] in which a linear relationship between

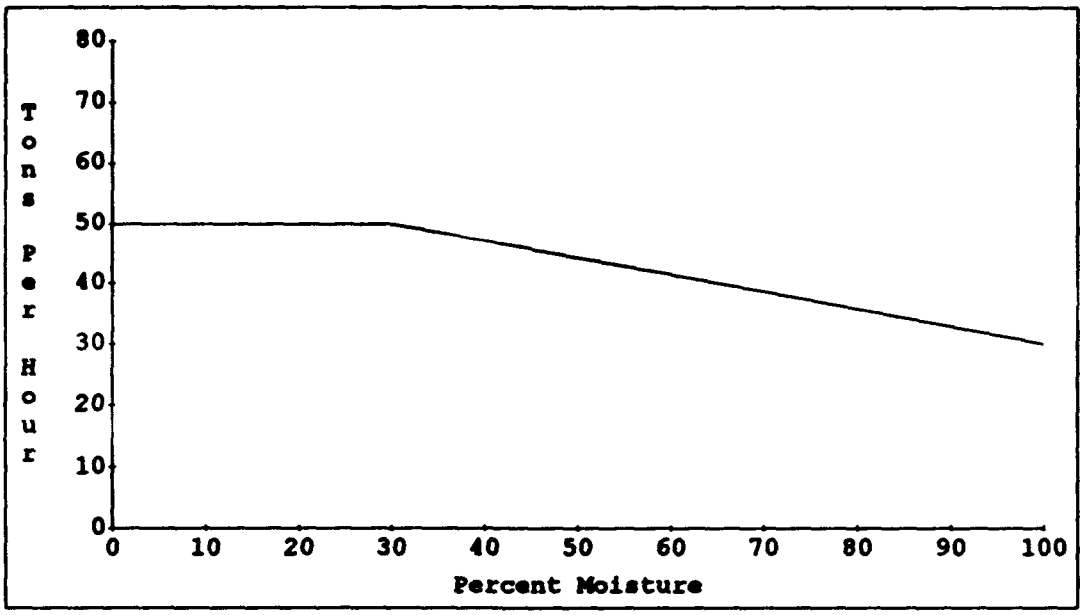


Figure 7. Dryer Flow Rate Versus Input Sawdust Moisture

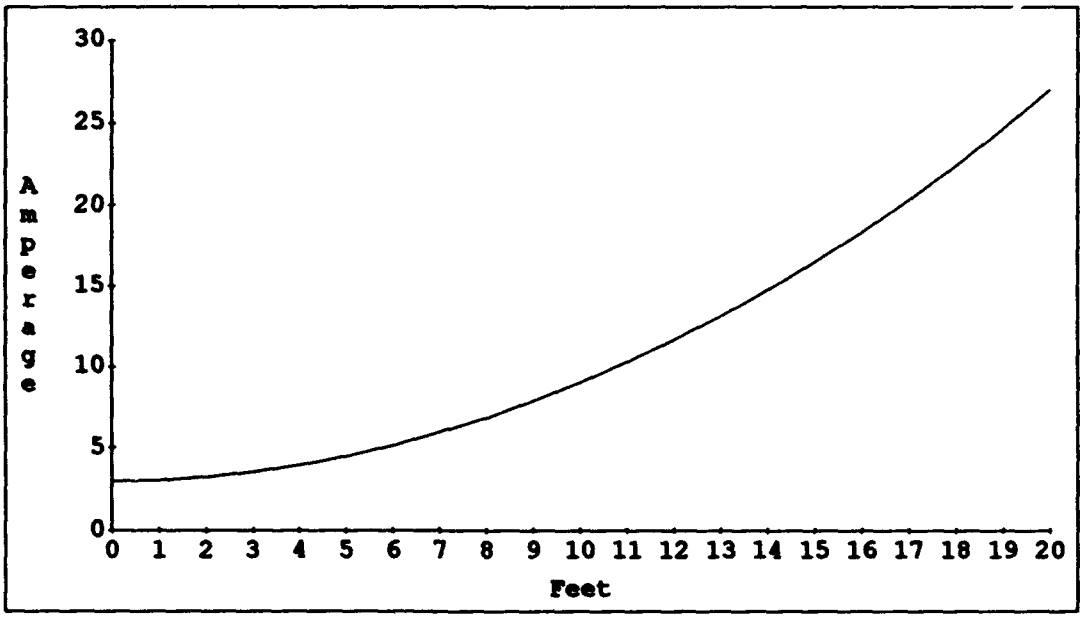


Figure 8. Discharge Screw Amps Versus Surge Bin Level

percent moisture in wood and the energy required to evaporate this water is shown.

The surge bin discharge screw is the conveying system that moves the sawdust from the bin to the dryer. This system is modeled so that higher bin sawdust levels cause the screw to work harder. That is, the force applied to turn the blades of the screw is resisted by the resultant force of the sawdust piled on top of the screw, so higher bin levels result in greater resistance and higher discharge screw amperage readings (Figure 8).

The temperature of the dryer is maintained by a fan/damper system that pulls hot air through the dryer. The fan has five speed settings where speed 1 is the slowest (coolest setting) and speed 5 is the fastest (hottest setting). The maintainable temperatures for each of these speeds at a given input sawdust moisture content is shown in Figure 9. Since higher fan speeds use more energy to draw greater volumes of hot air through the dryer, the lowest fan setting that can maintain the required 215-225°F temperature range is desirable. An open-air damper with the characteristics shown in Figure 10 aids in keeping the fan set as low as possible. The damper is automatically positioned so that its ±5° effect on dryer temperatures is applied as to keep the fan at its lowest possible energy setting (lowest speed) where proper dryer temperatures are achievable.

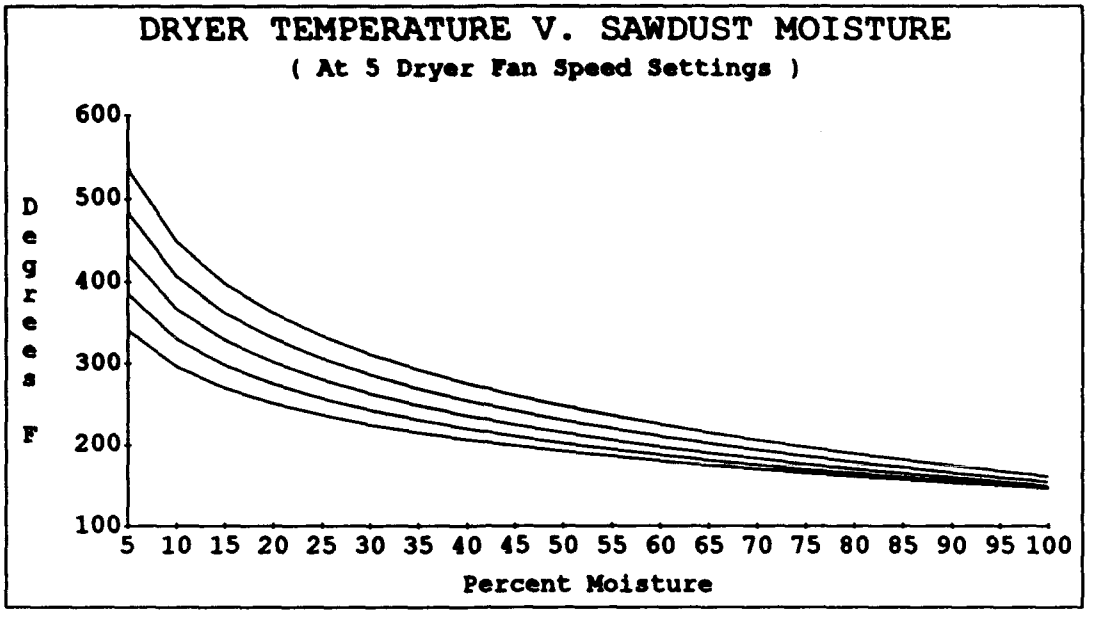


Figure 9. Dryer Fan Speed Curves

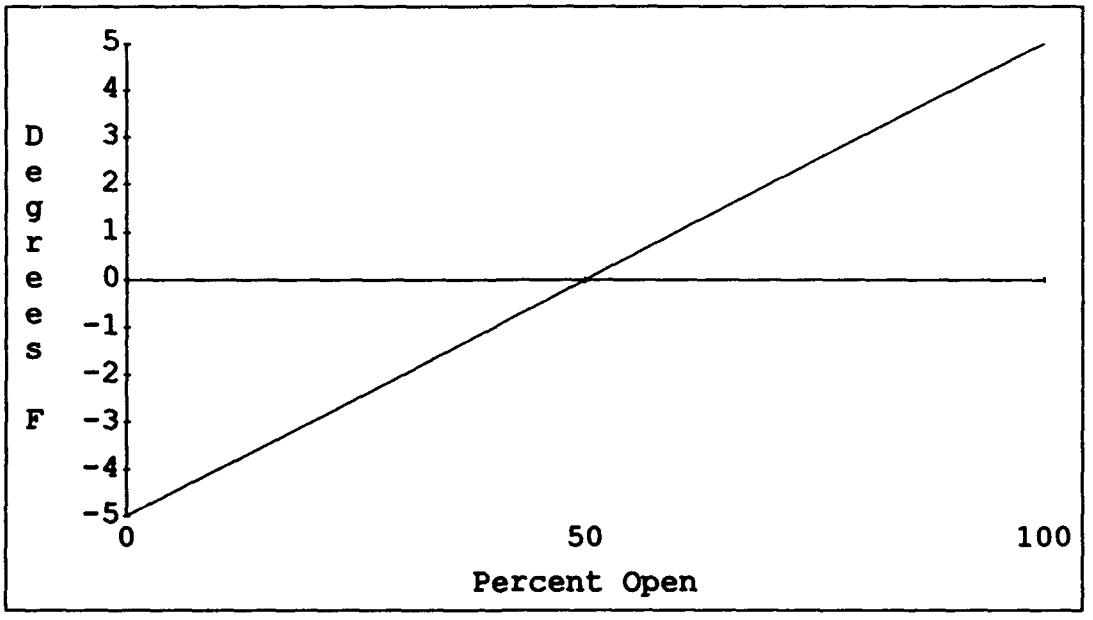


Figure 10. Damper Position Versus Temperature Change

The output sawdust moisture is a function of the temperature the dryer is able to maintain (Figure 11). This assumes the ideal operation of the dryer flow rate regulator.

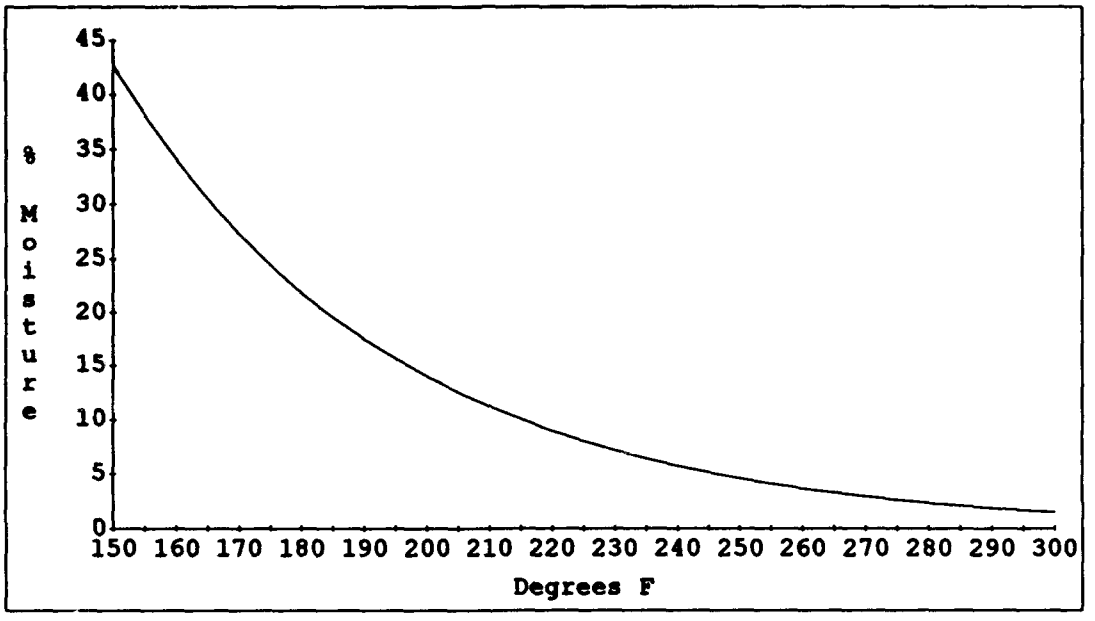


Figure 11. Dryer Temperature Versus Output Moisture Level

Based on the proceeding process parameter relationships, DRYER-SIM provides a non-linear test system for the expert network to analyze and control. In addition to normal dryer operations, DRYER-SIM also simulates a number of different system fault situations. These include numerous level sensor failure combinations, stuck damper, extremely high sawdust input moisture, drag chain failure, discharge screw failure, and fire. In this way, various

fault/process parameter combinations are available for simulation in order to provide a more thorough test bed.

Finally, DRYER-SIM can be run with the expert network feedback disconnected from the system. This feature enables the viewing of how various faults and parameters settings affect dryer operations without a system analyzer/controller in the loop (see APPENDIX B for DRYER-SIM source code).

2. Network Architecture. The prototype expert network (PEN) is designed to monitor DRYER-SIM and report any deviations from normal operations. Also, the PEN is tasked with operating the drag chain in order to keep the surge bin adequately filled (like in [15]) and controlling the fan/damper system to keep dryer temperatures within the permissible range at the lowest possible fan speed.

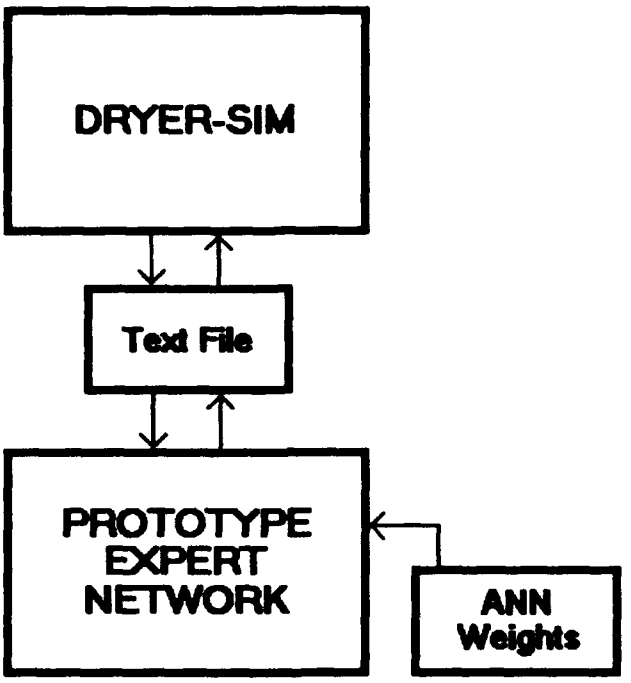


Figure 12. System Block Diagram

Unlike DRYER-SIM, the PEN is programmed in Pascal. The interface between the two programs (Figure 12) is accomplished with a bridging text file that contains sensor values when generated from DRYER-SIM and system state/control information when generated from the PEN. The process parameter sample period is 4 seconds with the PEN disabled and 10 seconds with the PEN in control. These periods are, of course, user seconds as each sample represents 5 minutes of simulated dryer operations.

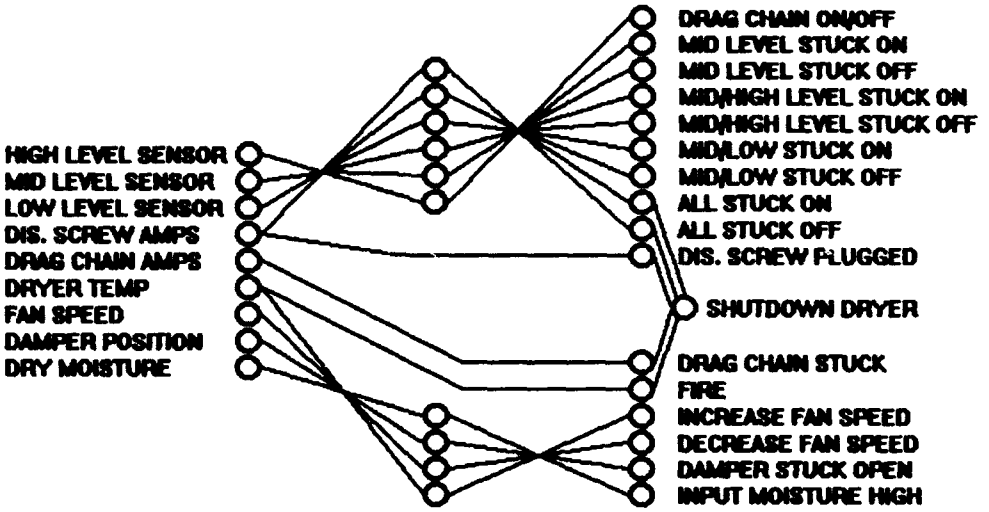


Figure 13. PEN Architecture

The PEN architecture (Figure 13) is fairly straight forward. Basically, it is composed of two feed forward, three layer backpropagation networks along with several independent neurons. The cross-links in Figure 13 represent fully connected layers; the direct attachments represent simple threshold connections. The PEN inputs are composed

of 9 sensor readings which are pre-filtered to form the discrete values listed in Table I. The PEN output nodes represent system states and control actions. Output values are sigmoid binary (0.1 and 0.9) where a high indicates an active neuron. For instance, if the MID LEVEL STUCK ON node has a high value (0.9), this particular systems fault is identified; a low value (0.1) indicates this fault is not present. The fault detection scheme is purely a classification problem where a particular error is diagnosed on the basis of how closely its known symptoms are characterized by the system input values [16].

The 4x6x9 network identifies surge bin sensor errors and, depending on the level of the bin and accounting for any stuck switches, decides whether or not the drag chain should be turned on or off in order to maintain the desired amount of sawdust in the bin. The 4x4x4 network controls the dryer temperature regulation system and determines two faults: stuck damper and input sawdust moisture too high. The simple threshold neurons identify three of the five critical faults: a stuck drag chain, plugged discharge screw, or fire in the dryer drum. The other two critical faults occur when the bin level sensors are all stuck on or off. The SHUTDOWN DRYER output is set by any activation of any of the critical fault neurons.

The number of hidden nodes selected for each of the two layered networks is based on training performance. That is, the 4x6x9 network converged to a solution quicker with 6

Table I. Input Layer Organization

NODE	SENSOR TYPE	THRESHOLDING
1	High Level Switch	on:0.9, off:0.1
2	Mid Level Switch	on:0.9, off:0.1
3	Low Level Switch	on:0.9, off:0.1
4	Discharge Screw Amps	[0,5) : 0.1 [5,15) : 0.4 [15,27): 0.6 [27+] : 0.9
5	Drag Chain Amps	>= 30: 0.9 < 30: 0.1
6	Dryer Temperature	> setpoint: 0.9 = setpoint: 0.5 < setpoint: 0.1
7	Dryer Fan Speed	speed 1: 0.1 speed 2: 0.3 speed 3: 0.5 speed 4: 0.7 speed 5: 0.9
8	Damper Position	> 50% open: 0.9 = 50% open: 0.5 < 50% open: 0.1
9	Dry Sawdust Moisture	> 10%: 0.9 8-10%: 0.5 < 10%: 0.1

hidden neurons while the 4x4x4 network seemed to learn faster when 4 middle layer nodes were used. Its possible that the quicker training results were unrelated to the number of hidden neurons, especially since each training session begins with randomized connection weights. Lacking

any established procedure for determining the optimal number of hidden nodes, however, this was the criteria used. In the end, both networks trained adequately with their respective hidden layer makeups as shown in the next section (see APPENDIX C for network source code).

IV. RESULTS AND DISCUSSION

A. NETWORK TRAINING

As discussed, one of the advantages of neural networks over traditional optimization approaches is the ability of properly trained networks to use learned input-output relationships to generalize functional associations for unfamiliar inputs. The individual networks contained in the PEN were progressively trained using as little information as possible to illustrate the generalization abilities of backpropagation based networks. The results showed that, at least for the PEN, only a fraction of a system's total input space need be covered in training in order to produce highly accurate networks (where accuracy is a measure of a network's ability to respond correctly to all input presentations). This is, obviously, of great significance for applications of high order.

The PEN uses 9 inputs to determine 1 of 59 possible DRYER-SIM states. Accounting for DRYER-SIM assumptions, program options, and input thresholding, there are a total of 1248 discrete combinations of the 9 inputs. Considering only the layered networks of the PEN, there are 8 inputs totaling 578 possible combinations. The 4x6x9 network (NET1) has 4 inputs that can take on 22 different value combinations, and the 4x4x4 network (NET2) has 4 inputs that total 41 possible value combinations (some of the NET1 and

NET2 combinations cannot occur at the same time resulting in the 578 combinations figure).

Both the randomness of pre-training weights and the selection of input-output vector exemplars affect a network's post-training accuracy. Because randomized weights place a network at a unique point on the error-weight surface at the start of every training attempt, a particular network can learn differently (better or worse) in two training sessions even when the same sample vectors are used. To more effectively evaluate how well the PEN learns, therefore, NET1 and NET2 were progressively trained and retrained five separate times. This involved some 315 individual training sessions that, combined, took nearly 53 computer hours to complete. Progressive training means using from none to all input-output associations (taken randomly) per session and examining the network performance at every stage. The results of each of the five sessions are averaged and shown in Figures 14 and 15 for NET1 and NET2 respectively.

Overall, NET1 did not average better than 95% accuracy until 17 of the 22 possible input-output associations were used. However, on its best pass, NET1 achieved 95% accuracy with 14 training vectors and 86% accuracy with only 10 vectors (Figure 16). Moreover, the three misclassifications for the 86% case occurred only in critical fault cases where the output nodes erroneously activated were already inhibited by the SHUTDOWN DRYER threshold neuron

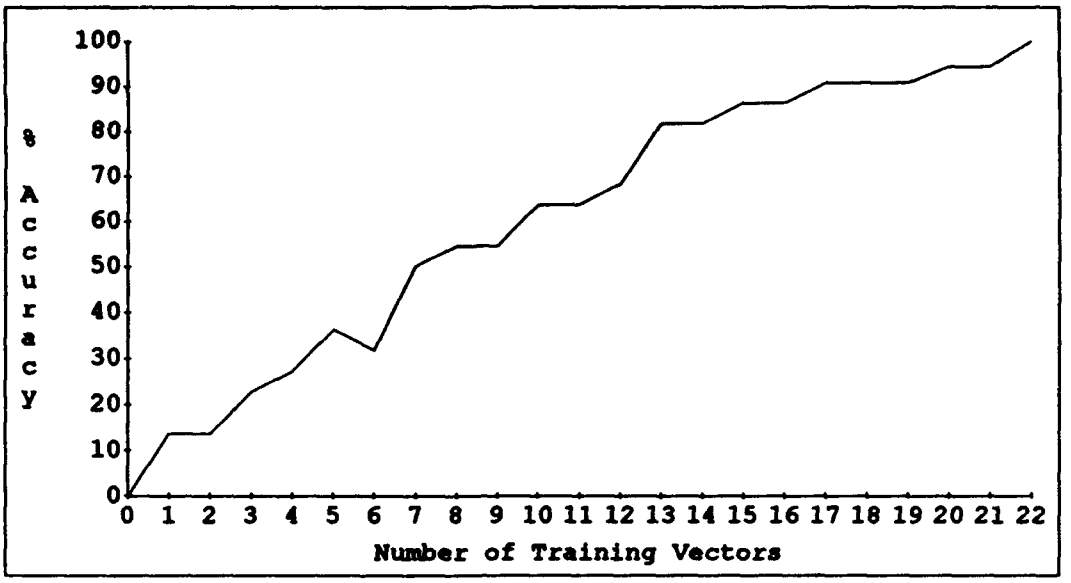


Figure 14. Average Progressive Training Results for NET1

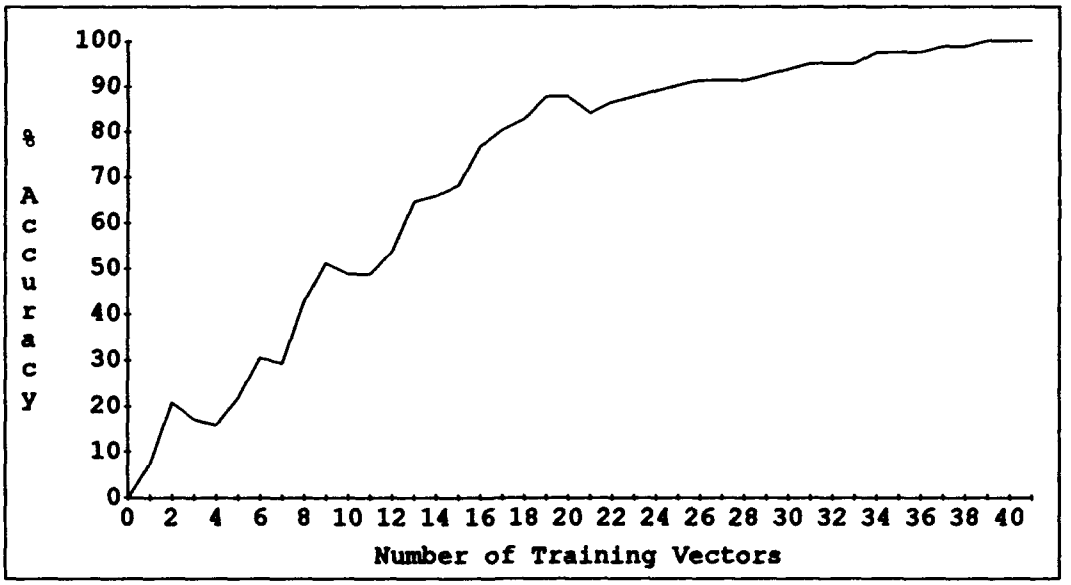


Figure 15. Average Progressive Training Results for NET2

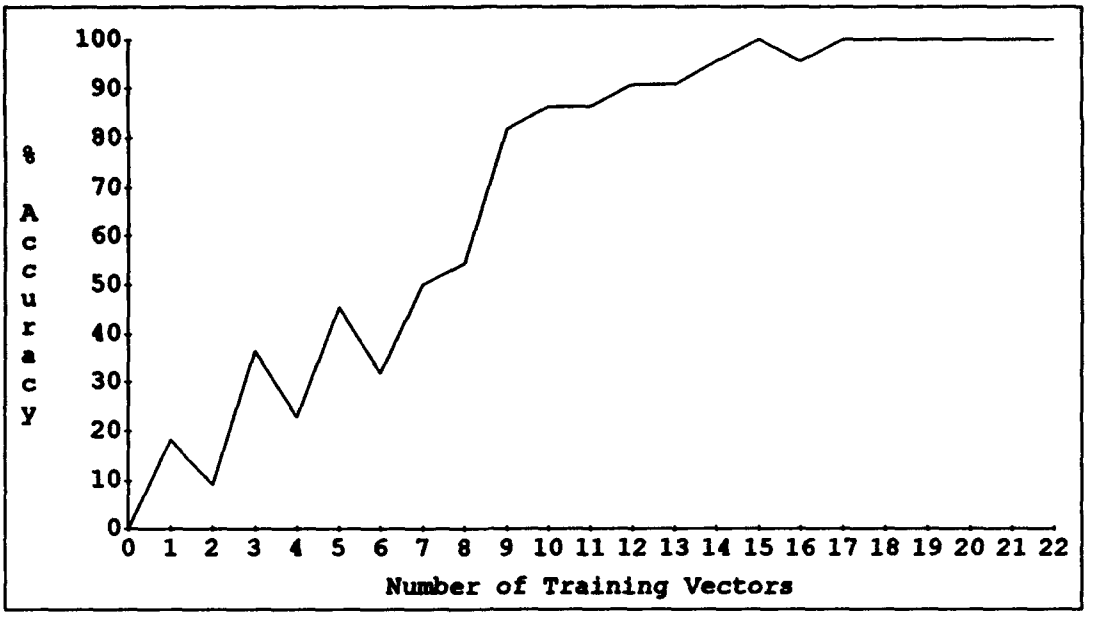


Figure 16. Best NET1 Progressive Training Session

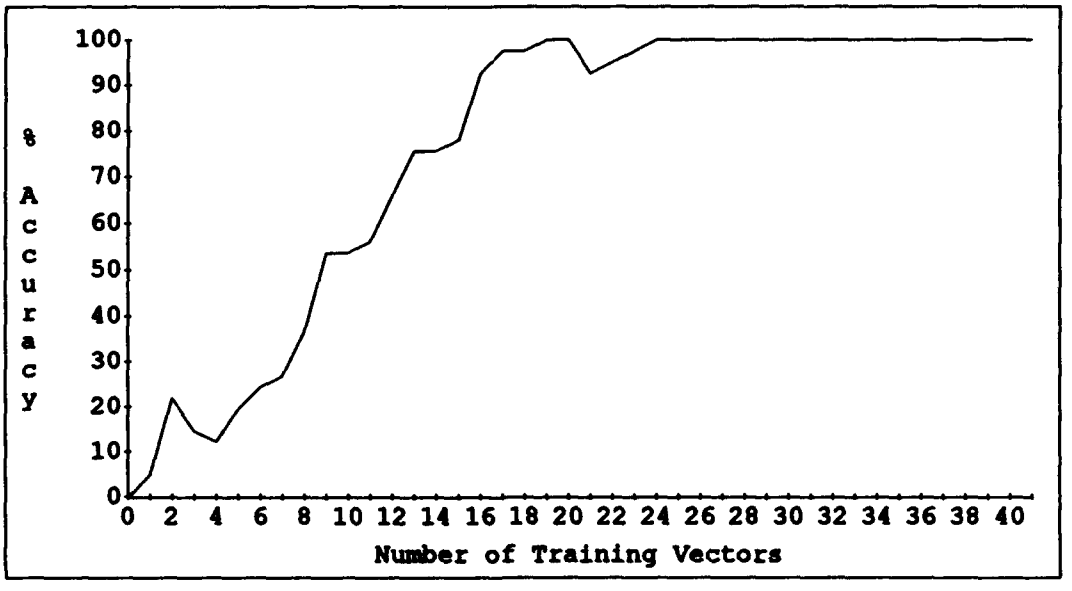


Figure 17. Best NET2 Progressive Training Session

(Figure 13). NET1 is therefore 100% effective at an accuracy of 86% trained over only 45% of its input space.

NET2 results are slightly better than that of NET1 in terms of network efficiency. Over the 5 training sessions, NET2 did not achieve 95% accuracy until an average of 31 of 41 input-output vectors were applied, or 76% of its input space. On its best pass, however, 100% accuracy was achieved with only 19 training samples (Figure 17). NET2, therefore, correctly classifies 100% of input-output associations it is only 46% familiar with (see APPENDIX D for best pass training data).

To summarize, there are 578 combinations of the 2 sets of 4 inputs that feed NET1 and NET2. These networks are trained over 45% (NET1) and 46% (NET2) of their respective input spaces. These percentages, however, combine to cover 118 of the 578 input possibilities. Discounting the single threshold input, therefore, the PEN is effectively 100% accurate, trained over a mere 20% of its input space. If the 9th input or independent thresholding node is considered, the PEN efficiency is slightly greater as a little less than 20% (247 of 1248) of all input-output patterns are explicitly used in development. When viewed as a whole, therefore, the PEN architecture demonstrates fairly efficient generalization abilities; a trait which is extremely advantageous for actual control applications where, as opposed to simulation, system models are often

larger in scope and less well defined because of real-world complexities.

B. PERFORMANCE

Since the PEN input-output associations are 100% accurate in terms of recognizing DRYER-SIM states, its control performance under simulation is, accordingly, 100% effective. In spite of the relatively small size of the process simulator, the degree of accuracy obtained by the incompletely trained PEN is noteworthy to say the least. It may, in fact, indicate that the neural networks are provided with too much information and a larger problem domain is required to reveal the actual limitations of this particular architecture. Whatever the case, the PEN certainly surpasses its design expectations in terms of performance and efficiency. Future considerations for even better PEN results and larger PEN based applications are presented in the next section (see APPENDIX E for training program source code).

V. CONCLUSIONS AND RECOMMENDATIONS

A. PROTOTYPE SYSTEM SUMMARY

The PEN's implementation success at the simulation level illustrates the promise such ANN base systems show for control applications. The processing of numeric, rather than symbolic, data gives the PEN a definite speed advantage over a comparable rule based system. PEN implementation in parallel hardware would only widen this speed margin. If DRYER-SIM process specifications change, PEN maintainability merely requires the encoding of any new sensor-response patterns. Because of a standardized training procedure, no thought must be given as to how new pattern(s) must be applied to the PEN in order to properly interrelate with sensor-response associations already in place; retraining procedures take care of this automatically. Finally, expert network implementations like the PEN are adept at determining input-output functional mappings when in unfamiliar problem domain territory, a quality not possessed by more conventional control methods.

B. ENHANCEMENT CONSIDERATIONS

While DRYER-SIM is by no means a trivial program, it could be expanded in terms of the number of process parameters and the complexity of the mathematical relationships that govern its operation. Rather than construct another model, however, the next logical step for

the retort monitor project is to apply to an expert network to an actual system of interest. Such a step will certainly introduce the real-world complexities lacking in the current software simulation.

With regards to programming efficiency, the PEN could (and most likely should) be reprogrammed in a language more compatible with LEVEL5's object oriented Production Rule Language which was used to create DRYER-SIM. LEVEL5 (version 2.0 +) is designed to interface with server programs written in Microsoft C (version 5.0 +). To circumvent this, DRYER-SIM to PEN communication is accomplished over a text file bridge, and the non-Windows, Turbo Pascal programmed PEN is timer activated and runs according to a Window's program information file. In this way, the DRYER-SIM is always an active application, and the PEN turns on and off at its designated sample times to make the required process control adjustments (if any). If the PEN were written in C, DRYER-SIM would run smoother as a single Window's application. Also, the fact that C is the industry standard language for engineering applications further increases the desirability of such a switch.

Finally, although the PEN training analysis presented in this thesis took literally days of computer time to accomplish, further analysis might result in even better network performance. When dealing with the still inexact science of training backpropagation based networks, exhaustive training sessions are the only way to gain an

accurate portrayal of a particular network's true abilities with regards to its problem domain.

C. FUTURE RESEARCH

Essentially, this thesis was a feasibility study for the development of an expert network based process monitor for an actual industrial application. Only briefly touched on, however, was the subject of knowledge acquisition to obtain actual process control data in the form of heuristics. Undoubtedly, compiling enough system information for a viable expert network is one of the most important and certainly the most difficult tasks left to be accomplished for the retort monitor project.

While the PEN proved effective for its level of implementation, a larger scale, real-world expert network application will almost certainly require more sophistication in terms of the particulars of the ANN architecture employed. There are numerous different network paradigms available, and many have eliminated a number of the shortcomings associated with backpropagation. Whether or not one is more appropriate for use in the final expert network construction will not be evident until much more is determined in terms of the final system requirements.

APPENDIX A

DRYER-SIM USER'S GUIDE

DRYER-SIM

USER'S GUIDE

INTRODUCTION

DRYER-SIM is a process simulator that mimics the operation of a wood dryer used in an industrial retort. Its purpose is to demonstrate the capabilities of an external neural network diagnostic / control module. To evaluate network performance, DRYER-SIM operations may be viewed with and without the neural net module operating. This way you can see how the system reacts to parameter changes with and without an active controller present. With DRYER-SIM, key process variables may be changed and various system faults may be introduced in order to more completely access the validity of the network control procedures.

SYSTEM REQUIREMENTS

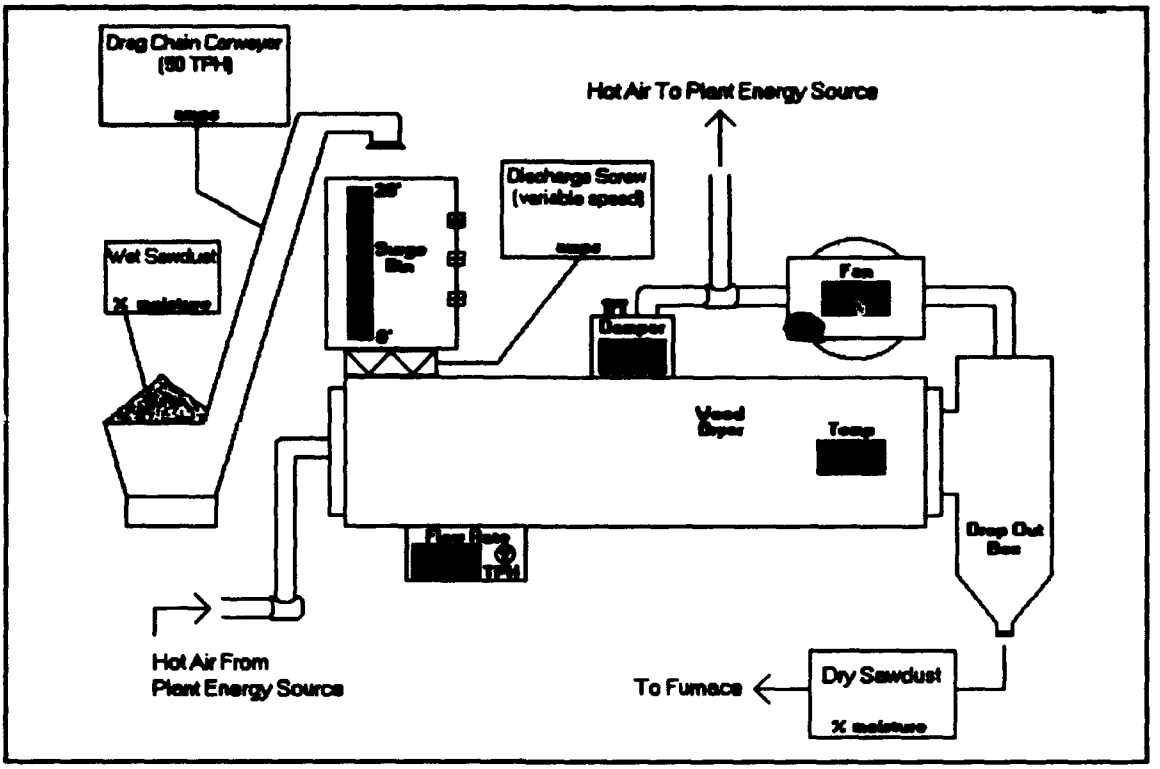
To run DRYER-SIM the following is required:

SOFTWARE:

- Microsoft Windows Version 3.1
- LEVEL5 OBJECT Version 2.5.1

HARDWARE:

- 80386 or greater microprocessor with at least 2MB of memory
- A hard disk drive with a minimum of 1.5MB free disk space
- VGA graphics board (1024x768 video driver recommended)
- A mouse

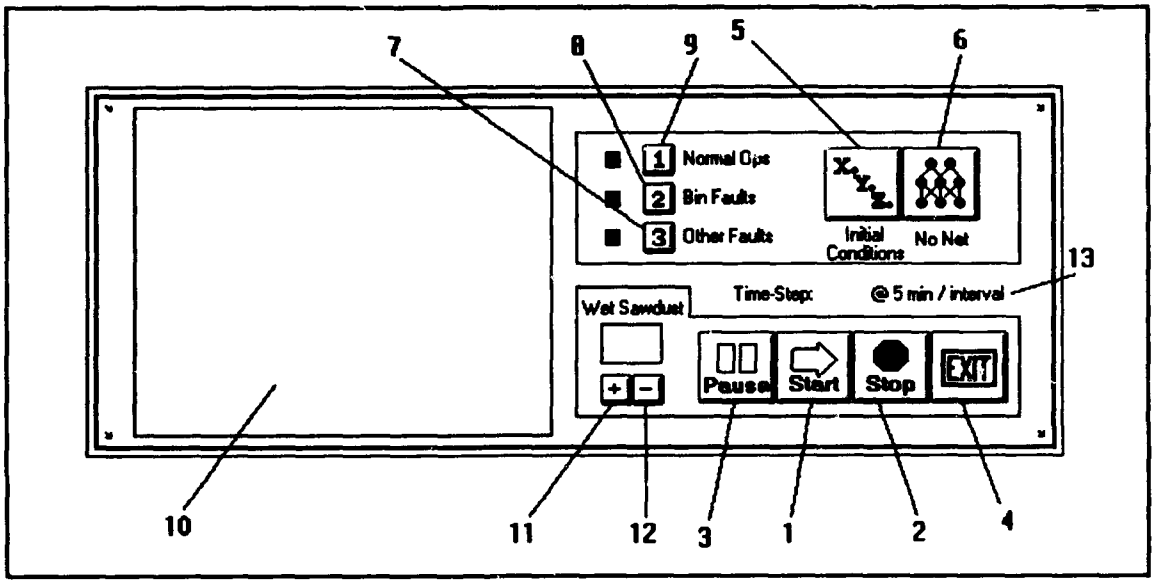


Dryer screen

DRYER-SIM OVERVIEW

The simulated dryer operations are fairly straightforward. Wet (high moisture) sawdust enters the system via the drag chain conveyer that operates in an on/off manner to keep the surge bin filled about the 10 foot (middle) level. There are three level sensors placed at the 5, 10, and 15 feet marks that keep track of the amount of sawdust in the bin. When a sensor is on, the sawdust level is at or above that sensor. Sawdust from the surge bin is fed to the dryer through the discharge screw at rate dependent on the amount of moisture in the sawdust. The dryer temperature setpoint is 220 ±5°F. High moisture sawdust takes longer to dry at 220°F, so the dryer runs slower when lower sawdust moisture is being processed. The dryer temperature is maintained by the fan that draws heat through the dryer from the plant-wide energy source. Higher fan speeds cause higher temperatures. Lower fan speeds have the opposite effect. Sawdust leaves the dryer through the drop out box at an 8-10% moisture level (assuming normal operations). Abnormal operations have various affects on the output sawdust moisture range.

There are four variables you have control over: Input sawdust range, initial feed bin level, and initial drag chain setting (on or off), and initial fan speed setting (1-5). The input sawdust range can be changed at any time during a simulation. The three other variables, bin level drag chain setting, and fan speed, can only be set before a simulation begins. Changes are affected via the control panel shown below. The function of each control is described next.



DRYER-SIM CONTROL PANEL

CONTROL PANEL

The numbers on the following list of items correspond to the numbered sections of the control panel figure above.

1. **Start Button** - Begins a simulation using the current initial conditions (bin level and drag chain setting) and system setting (normal operations, level faults, or other faults).

2. **Stop Button** - Ends the current simulation. Resets initial conditions that are out of the specifications for normal operations.

3. **Pause Button** - Pauses the current simulation. From pause the only remaining options are to unpause the current simulation, stop the current simulation, or exit DRYER-SIM.

4. **Exit Button** - Quits DRYER-SIM and returns to the LEVEL5 OBJECT environment.

5. **Initial Conditions Button** - This button enables you to set the value a dryer variable will have at the beginning of a simulation. Initial conditions can be set before any simulation, including those with faults. The three variables are:
   1. Bin Level: select from 3 to 20 feet
   2. Drag Chain: select on or off
   3. Fan Speed: select from speeds 1(slowest) to 5(fastest)

6. **No Net Button** - This button toggles between "No Net" and "Network" depending on its current setting. Set to No Net, the neuro-controller module is disabled and the system is without an active controller. With this setting, the drag chain operates depending on the value of the middle level sensor. If the sensor is on the drag chain stops; if its off the drag chain starts. If there are faults that affect the middle sensor, the bin level will not be maintained. The Network setting enables the neuro-controller which assumes control of the drag chain operation to keep the bin filled and fan settings to keep the dryer at the proper temperature. The Net/No Net option is available for all simulations, including those with faults.

7. **Normal Operations Button** - This button sets all system parameters to standard (no fault) operations and clears any previous fault settings See **Normal Operations** section for further information).

8. **Bin Faults Button** - This button triggers a pop-up window in the auxiliary control display (10). In the window is a menu where you can select one of eight different bin sensor failures situations (See **Surge Bin Faults** section for further information).

9. **Other Faults Button** - This button triggers a pop-up window in the auxiliary control display (10) that presents a menu where you can select one of 5 different system fault situations (See **Other Faults** section for further information).

10. **Auxiliary Control Display** - This is a display area of the control panel where various menus are displayed and selections are made.

11. **Increase Wet Sawdust Moisture** - This button augments the moisture content of the input sawdust. The allowed range is from 30% to 70%. When the high input moisture fault is set the top value is extended to 90%.

12. **Decrease Wet Sawdust Moisture** - This button lessens the moisture content of the input sawdust. The allowed range is from 30% to 70%. When the high input moisture fault is set the top value is extended to 90%.

13. **Current Time Interval Display** - This section of the control panel displays the sample period from the start of the current simulation. Each iteration corresponds to five minutes of dryer operations. With the neuro-controller disabled an iteration occurs every 4 seconds. Iterations occur every 10 seconds under neuro-control.

NORMAL OPERATIONS

Under normal operation procedures, no system faults occur and all process parameters are computed according to their particular specifications. Under No Net normal operations, the system will function properly as long as the input sawdust moisture is not changed to where the dryer temperature cannot be maintain at the current fan setting (remember there is no controller to reset the fan speed under No Net operations).

SURGE BIN FAULTS

There are eight sensor errors that can be set when bin fault operations is selected. These faults are:

1. Mid Level Switch Stuck On
2. Mid Level Switch Stuck Off
3. Mid and High Level Switch Stuck On
4. Mid and High Level Switch Stuck Off
5. Mid and Low Level Switch Stuck On
6. Mid and Low Level Switch Stuck Off
7. All Level Switches Stuck On
8. All Level Switches Stuck Off

The meaning of each of these faults is obvious. When a bin fault is set along with No Net control, the bin level cannot be maintained because the middle sensor is always affected. Under neuro-control, however, the network recognizes which fault has occurred and compensates to keep the bin filled

based on a working sensor (except in the case of all switches failing where the network automatically shuts down the dryer). Other faults cannot be set concurrently with level sensor failures and selecting normal operations clears all fault settings.

OTHER FAULTS

There are five system faults that can be set with this options. These are:

1. Drag Chain Stuck
2. Discharge Screw Plugged
3. High Input Sawdust Moisture
4. Stuck Damper
5. Fire

When the drag chain is stuck, the surge bin is not refilled. When the discharge screw plugs (becomes stopped up with debris) the dryer receives no sawdust input. When extremely high input sawdust moisture is allowed, the permissible range for the input sawdust variable is from 30% to 90%. This variable can be set at any point in a simulation. A stuck damper does not always aid in keeping the dryer fan at the lowest possible speed setting that maintains proper drying temperatures. A fire is characterized by a rapid increase in the temperature of the dryer. Again, these faults are only detectable under neuro-control. The neuro-controller shuts down the dryer when it detects a stuck drag chain, plugged screw, or fire. Bin level faults cannot be set along with system faults and selecting normal operations clears all fault settings.

APPENDIX B

DRYER-SIM SOURCE CODE

```
$VERSION25
$LOCATIONS ARE PIXELS


ATTRIBUTE computer screen PICTURE
  ARRAY SIZE 40
ATTRIBUTE start computer SIMPLE
ATTRIBUTE frame number NUMERIC
ATTRIBUTE begin SIMPLE
ATTRIBUTE total frames NUMERIC
ATTRIBUTE start simulation SIMPLE


INSTANCE the domain ISA domain
  WITH computer screen [1 ] := "L5G00000.bmp"
  WITH computer screen [2 ] := "L5G00001.bmp"
  WITH computer screen [3 ] := "L5G00002.bmp"
  WITH computer screen [4 ] := "L5G00003.bmp"
  WITH computer screen [5 ] := "L5G00004.bmp"
  WITH computer screen [6 ] := "L5G00005.bmp"
  WITH computer screen [7 ] := "L5G00006.bmp"
  WITH computer screen [8 ] := "L5G00007.bmp"
  WITH computer screen [9 ] := "L5G00008.bmp"
  WITH computer screen [10 ] := "L5G00009.bmp"
  WITH computer screen [11 ] := "L5G00010.bmp"
  WITH computer screen [12 ] := "L5G00011.bmp"
  WITH computer screen [13 ] := "L5G00012.bmp"
  WITH computer screen [14 ] := "L5G00013.bmp"
  WITH computer screen [15 ] := "L5G00014.bmp"
  WITH computer screen [16 ] := "L5G00015.bmp"
  WITH computer screen [17 ] := "L5G00016.bmp"
  WITH computer screen [18 ] := "L5G00017.bmp"
  WITH computer screen [19 ] := "L5G00018.bmp"
  WITH computer screen [20 ] := "L5G00019.bmp"
  WITH computer screen [21 ] := "L5G00020.bmp"
  WITH computer screen [22 ] := "L5G00021.bmp"
  WITH computer screen [23 ] := "L5G00022.bmp"
  WITH computer screen [24 ] := "L5G00023.bmp"
  WITH computer screen [25 ] := "L5G00024.bmp"
  WITH computer screen [26 ] := "L5G00025.bmp"
  WITH computer screen [27 ] := "L5G00026.bmp"
  WITH computer screen [28 ] := "L5G00027.bmp"
  WITH computer screen [29 ] := "L5G00028.bmp"
  WITH computer screen       := "L5G00029.bmp"
  WITH computer screen [     := "L5G00030.bmp"
  WITH computer screen [32 ] := "L5G00031.bmp"
  WITH computer screen [33 ] := "L5G00032.bmp"
  WITH computer screen [34 ] := "L5G00033.bmp"
```

```
  WITH computer screen [35 ] := "L5G00034.bmp"
  WITH computer screen [36 ] := "L5G00035.bmp"
  WITH computer screen [37 ] := "L5G00036.bmp"
  WITH computer screen [38 ] := "L5G00037.bmp"
  WITH computer screen [39 ] := "L5G00038.bmp"
  WITH computer screen [40 ] := "L5G00039.bmp"
  WITH start computer := TRUE
  WITH frame number := 1
  WITH total frames := 40


INSTANCE the application ISA application
  WITH unknowns fail := TRUE
  WITH threshold := 50
  WITH title display := title screen
  WITH ignore breakpoints := FALSE
  WITH reasoning on := FALSE
  WITH numeric precision := 8
  WITH simple query text := "Is it true that:
  *
is
  *"
  WITH numeric query text := "What is(are):
  *
of
  *"
  WITH string query text := "What is(are):
  *
of
  *"
  WITH [illegible] query text := "What is(are):
  *
of
  *"
  WITH interval query text := "What is(are):
  *
of
  *"
  WITH compound query text := "What is(are):
  *
of
  *"
  WITH multicompound query text := "What is(are):
  *
of
  *"
  WITH demon strategy IS fire first
  WITH visible file menu := FALSE
```

```
INSTANCE title screen ISA display
  WITH wait := FALSE
  WITH delay changes := FALSE
  WITH items [1 ] := textbox 9
  WITH items [2 ] := picturebox 1
  WITH items [3 ] := picturebox 2
  WITH items [4 ] := pushbutton 1
  WITH items [5 ] := textbox 11
  WITH items [6 ] := pushbutton 2
  WITH items [7 ] := textbox 10


INSTANCE system info ISA display
  WITH wait := FALSE
  WITH delay changes := FALSE
  WITH items [1 ] := textbox 13
  WITH items [2 ] := pushbutton 3
  WITH items [3 ] := textbox 14
  WITH items [4 ] := UNDETERMINED
  WITH items [5 ] := UNDETERMINED
  WITH items [6 ] := picturebox 3
  WITH items [7 ] := textbox 15
  WITH items [8 ] := textbox 16
  WITH items [9 ] := textbox 17
  WITH items [10 ] := textbox 18
  WITH items [11 ] := textbox 19
  WITH items [12 ] := textbox 20
  WITH items [13 ] := picturebox 4
  WITH items [14 ] := UNDETERMINED
  WITH items [15 ] := textbox 21
  WITH items [16 ] := textbox 22
  WITH items [17 ] := textbox 23
  WITH items [18 ] := textbox 24
  WITH items [19 ] := textbox 12


INSTANCE picturebox 1 ISA picturebox
  WITH location := 695,60,930,305
  WITH clipped := TRUE
  WITH frame := FALSE
  WITH picture := "L5G00040.bmp"


INSTANCE picturebox 2 ISA picturebox
  WITH location := 743,95,880,176
  WITH clipped := TRUE
  WITH frame := FALSE
  WITH picture := "L5G00041.bmp"
```

```
INSTANCE picturebox 3 ISA picturebox
  WITH location := 55,345,115,690
  WITH clipped := TRUE
  WITH filename := "C:\\PROJECT\\ROW.BMP"


INSTANCE picturebox 4 ISA picturebox
  WITH location := 555,345,590,445
  WITH clipped := TRUE
  WITH filename := "C:\\PROJECT\\ROW2.BMP"


INSTANCE pushbutton 1 ISA pushbutton
  WITH location := 205,540,400,630
  WITH label := "START SIMULATION"
  WITH attribute attachment := start simulation OF the
domain


INSTANCE pushbutton 2 ISA pushbutton
  WITH location := 635,540,830,630
  WITH label := "PROGRAM OVERVIEW"
  WITH display attachment := system info


INSTANCE pushbutton 3 ISA pushbutton
  WITH location := 680,640,830,705
  WITH label := "START SIMULATION"
  WITH attribute attachment := start simulation OF the
domain


INSTANCE textbox 9 ISA textbox
  WITH location := 25,25,1000,715
  WITH justify IS left
  WITH font := "System"
  WITH frame := TRUE
  WITH text := ""


INSTANCE textbox 10 ISA textbox
  WITH location := 95,60,625,190
  WITH justify IS center
  WITH font := "MadAve"
  WITH font style IS bold, italic CF FALSE, underline,
strikeout CF FALS\
E
  WITH font size := 55
  WITH text := "DRYER-SIM"
```

```
INSTANCE textbox 11 ISA textbox
  WITH location := 100,240,620,395
  WITH justify IS center
  WITH font := "MS Sans Serif"
  WITH font style IS bold CF FALSE, italic CF FALSE,
underline CF FALSE,\
 strikeout CF FALSE
  WITH font size := 14
  WITH text := "A Computer Simulation for an Expert Network
Application

Douglas M. Rogers
1994"


INSTANCE textbox 12 ISA textbox
  WITH location := 230,50,805,115
  WITH justify IS center
  WITH font := "MadAve"
  WITH font style IS bold, italic CF FALSE, underline,
strikeout CF FALS\
E
  WITH font size := 24
  WITH text := "DRYER-SIM OVERVIEW"


INSTANCE textbox 13 ISA textbox
  WITH location := 25,25,1000,715
  WITH justify IS left
  WITH font := "System"
  WITH frame := TRUE
  WITH text := ""


INSTANCE textbox 14 ISA textbox
  WITH location := 185,130,850,295
  WITH pen color := 0,0,0
  WITH fill color := 0,255,255
  WITH justify IS left
  WITH font := "System"
  WITH frame := TRUE
  WITH text := "     This program simulates the operation of
a wood wast\
e dryer used in an industrial retort that produces charcoal.
The goal o\
f the simulator is to demonstrate the capabilities of an
external neural\
 network diagnostic / control module. To evaluate network
performance, d\
```

```
ryer system operations may be viewed with and without the
neural net mod\
ule operating.  Also, key process parameters may be changed
and various \
system faults may be set in order to further verify network
control proc\
edures and recommendations. The major simulation options are
summarized \
below."


INSTANCE textbox 15 ISA textbox
  WITH location := 120,360,425,395
  WITH justify IS left
  WITH font := "Small Fonts"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 7
  WITH text := "START:   Begins a simulation using the
current
     initial conditions and error options"


INSTANCE textbox 16 ISA textbox
  WITH location := 120,420,425,455
  WITH justify IS left
  WITH font := "Small Fonts"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 7
  WITH text := "STOP:    Ends the current simulation"


INSTANCE textbox 17 ISA textbox
  WITH location := 120,478,425,513
  WITH justify IS left
  WITH font := "Small Fonts"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 7
  WITH text := "PAUSE:   Freezes the current simulation"


INSTANCE textbox 18 ISA textbox
  WITH location := 120,535,485,570
  WITH justify IS left
  WITH font := "Small Fonts"
```

```
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 7
  WITH text := "IC: (Initial Conditions) Sets the surge bin
level and
     dryer fan speed startup settings"


INSTANCE textbox 19 ISA textbox
  WITH location := 120,591,485,626
  WITH justify IS left
  WITH font := "Small Fonts"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 7
  WITH text := "NET:      This option turns the neural
network control
     module off / on"


INSTANCE textbox 20 ISA textbox
  WITH location := 120,650,485,685
  WITH justify IS left
  WITH font := "Small Fonts"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 7
  WITH text := "EXIT:     Ends program - Returns user to
LEVEL5 object
     environment"


INSTANCE textbox 21 ISA textbox
  WITH location := 615,350,955,380
  WITH justify IS left
  WITH font := "Small Fonts"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 7
  WITH text := "NORMAL OPS:   Simulates normal system
operations
          with no system faults"


INSTANCE textbox 22 ISA textbox
  WITH location := 615,385,955,415
```

```
  WITH justify IS left
  WITH font := "Small Fonts"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 7
  WITH text := "LEVEL FAULTS: Simulates dryer operations
where
          various level sensors fail"


INSTANCE textbox 23 ISA textbox
  WITH location := 615,420,955,450
  WITH justify IS left
  WITH font := "Small Fonts"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 7
  WITH text := "OTHER FAULTS: Simulates dryer operations
where
          various system faults occur"


INSTANCE textbox 24 ISA textbox
  WITH location := 530,470,970,630
  WITH justify IS left
  WITH font := "Small Fonts"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 7
  WITH text := "PROCESS VARIABLES:

1. Wet Sawdust Moisture - % moisture of the wood waste
entering the drye\
r through the surge bin.  This is the main process variable
and is adjus\
table throughout all simulations.

2. Surge Bin Level - height (in feet) of the wood waste
collected in the\
 surge bin.  This parameter is adjustable as an initial
condition.

3. Dryer Fan Speed - Speed setting (1-5) of the dryer draft
fan.  This p\
arameter is adjustable as an initial condition."
```

```
INSTANCE computer timer ISA timer
  WITH period := 0 00:00:04.000


INSTANCE animation timer ISA timer
  WITH period := 0 00:00:02.500


INSTANCE main window ISA window
  WITH location := -1,-1,-1,-1
  WITH full screen := TRUE
  WITH style IS moveable CF FALSE, sizeable CF FALSE,
closeable CF FALSE
  WITH title := "DRYER SIM"
  WITH visible := TRUE
  WITH visible OK button := FALSE


INSTANCE expand window ISA window
  WITH location := -1,-1,-1,-1
  WITH full screen := TRUE
  WITH style IS moveable CF FALSE, sizeable CF FALSE,
closeable CF FALSE
  WITH title := ""
  WITH visible := TRUE
  WITH visible OK button := FALSE


DEMON for startup
IF start computer
THEN active OF computer timer
AND start OF computer timer := NOW


DEMON for go
IF begin
THEN picture OF picturebox 2 := computer screen[ frame
number]
AND frame number := (frame number MOD total frames) + 1
AND begin := begin


DEMON for simulation start
IF start simulation
THEN CHAIN "dryersim.knb"


DEMON for change screen
IF tripped OF computer timer
THEN picture OF picturebox 2 := computer screen[ 1]
```

```
AND location OF picturebox 2 := 742,93,880,176
AND active OF computer timer := FALSE
AND active OF animation timer := TRUE
AND start OF animation timer := NOW


DEMON for animation
IF tripped OF animation timer
THEN active OF animation timer := FALSE
AND begin

END




$VERSION25
$LOCATIONS ARE PIXELS

CLASS dryer surge bin
  WITH high level sensor SIMPLE
  WITH mid level sensor SIMPLE
  WITH low level sensor SIMPLE
  WITH level NUMERIC
  WITH init bin level up SIMPLE
    WHEN CHANGED
      BEGIN
        IF level OF surge bin < 20 THEN
          level OF surge bin := level OF surge bin + 1
        location OF bin level := SETRECT( 171, 216 - (level
OF surge bin\
 / 20 * 97), 183, 216)
      END
  WITH init bin level down SIMPLE
    WHEN CHANGED
      BEGIN
        IF level OF surge bin > 3 THEN
          level OF surge bin := level OF surge bin - 1
        location OF bin level := SETRECT( 171, 216 - (level
OF surge bin\
 / 20 * 97), 183, 216)
      END


INSTANCE surge bin ISA dryer surge bin
  WITH level := 13

CLASS dryer unit
  WITH flow rate NUMERIC
```

```
  WITH temperature NUMERIC
  WITH fan amps NUMERIC
  WITH damper position NUMERIC
  WITH fan rate NUMERIC
    ARRAY SIZE 5
  WITH fan index NUMERIC
  WITH damper temp change NUMERIC
  WITH init fan index up SIMPLE
    WHEN CHANGED
      BEGIN
        IF fan index OF the dryer < 5 THEN
          fan index OF the dryer := fan index OF the dryer +
1
        fan amps OF the dryer := fan index OF the dryer * 6
      END
  WITH init fan index down SIMPLE
    WHEN CHANGED
      BEGIN
        IF fan index OF the dryer > 1 THEN
          fan index OF the dryer := fan index OF the dryer -
1
        fan amps OF the dryer := fan index OF the dryer * 6
      END


INSTANCE the dryer ISA dryer unit
  WITH fan amps := 18
  WITH fan rate [1 ] := 285
  WITH fan rate [2 ] := 300
  WITH fan rate [3 ] := 315
  WITH fan rate [4 ] := 330
  WITH fan rate [5 ] := 345
  WITH fan index := 3

CLASS feed system
  WITH amps NUMERIC
  WITH on NUMERIC


INSTANCE drag chain conveyor ISA feed system
  WITH amps := 0
  WITH on := 0


INSTANCE dryer discharge screw ISA feed system

CLASS pictbtn INHERITS add on
  WITH location RECTANGLE
  WITH picture PICTURE
  WITH pressed picture PICTURE
```

```
  WITH disabled picture PICTURE
  WITH focus picture PICTURE
  WITH selected SIMPLE
  WITH attachment ATTRIBUTE_REFERENCE
  WITH enabled SIMPLE


INSTANCE start button ISA pictbtn
  WITH location := 440,655,489,702
  WITH picture := "L5G00042.bmp"
  WITH pressed picture := "L5G00043.bmp"
  WITH disabled picture := "L5G00044.bmp"
  WITH attachment := start OF the domain
  WITH enabled := TRUE


INSTANCE stop button ISA pictbtn
  WITH location := 490,655,539,702
  WITH picture := "L5G00045.bmp"
  WITH pressed picture := "L5G00046.bmp"
  WITH disabled picture := "L5G00047.bmp"
  WITH attachment := stop OF the domain
  WITH enabled := FALSE


INSTANCE exit button ISA pictbtn
  WITH location := 540,655,589,702
  WITH picture := "L5G00048.bmp"
  WITH pressed picture := "L5G00049.bmp"
  WITH attachment := exit OF the domain
  WITH enabled := TRUE


INSTANCE normal operations ISA pictbtn
  WITH location := 365,535,385,555
  WITH picture := "L5G00050.bmp"
  WITH pressed picture := "L5G00051.bmp"
  WITH disabled picture := "L5G00052.bmp"
  WITH attachment := select normal operations OF the domain
  WITH enabled := TRUE


INSTANCE level faults ISA pictbtn
  WITH location := 365,560,385,580
  WITH picture := "L5G00053.bmp"
  WITH pressed picture := "L5G00054.bmp"
  WITH disabled picture := "L5G00055.bmp"
  WITH attachment := select level switch failures OF the
domain
  WITH enabled := TRUE
```

```
INSTANCE other faults ISA pictbtn
  WITH location := 365,585,385,605
  WITH picture := "L5G00056.bmp"
  WITH pressed picture := "L5G00057.bmp"
  WITH disabled picture := "L5G00058.bmp"
  WITH attachment := select other failures OF the domain
  WITH enabled := TRUE


INSTANCE init cond ISA pictbtn
  WITH location := 475,535,524,582
  WITH picture := "L5G00059.bmp"
  WITH pressed picture := "L5G00060.bmp"
  WITH disabled picture := "L5G00061.bmp"
  WITH attachment := select initial conditions OF the domain
  WITH enabled := TRUE


INSTANCE control ISA pictbtn
  WITH location := 525,535,574,582
  WITH picture := "L5G00062.bmp"
  WITH pressed picture := "L5G00063.bmp"
  WITH disabled picture := "L5G00064.bmp"
  WITH attachment := select control OF the domain
  WITH enabled := TRUE


INSTANCE pause button ISA pictbtn
  WITH location := 390,655,439,702
  WITH picture := "L5G00065.bmp"
  WITH pressed picture := "L5G00066.bmp"
  WITH disabled picture := "L5G00067.bmp"
  WITH attachment := pause OF the domain
  WITH enabled := FALSE

CLASS sawdust
  WITH moisture NUMERIC
  WITH moisture up SIMPLE
    WHEN CHANGED
      BEGIN
        IF (moisture OF wet sawdust < 70 OR (use other AND
Other Failure\
s IS Enable High Moisture Range AND moisture OF wet sawdust
< 90)) THEN
          moisture OF wet sawdust := moisture OF wet sawdust
+ 1
      END
  WITH moisture down SIMPLE
```

```
    WHEN CHANGED
      BEGIN
        IF moisture OF wet sawdust > 30 THEN
          moisture OF wet sawdust := moisture OF wet sawdust
- 1
      END

INSTANCE wet sawdust ISA sawdust
  WITH moisture := 47

INSTANCE dry sawdust ISA sawdust

ATTRIBUTE start SIMPLE
ATTRIBUTE stop SIMPLE
ATTRIBUTE exit SIMPLE
ATTRIBUTE compute values SIMPLE
  WHEN CHANGED
    BEGIN
      IF (level OF surge bin = 0) OR (use other AND Other
Failures IS Di\
scharge Screw Plugged) THEN
        flow rate OF the dryer := 0
      ELSE
        flow rate OF the dryer := -0.28571 * moisture OF wet
sawdust + 5\
8.57143
      level OF surge bin := level OF surge bin + 0.003 *
0.08333 * (1000\
0 * drag chain conveyor.on - 200 * the dryer.flow rate)
      IF level OF surge bin > 20 THEN
        level OF surge bin := 20
      IF level OF surge bin < 0 THEN
        level OF surge bin := 0
      location OF bin level := SETRECT( 171, 216 - (level OF
surge bin /\
 20 * 97), 183, 216)
      IF level OF surge bin >= 10 THEN
        BEGIN
          IF (use bin level AND (Level Switch Failures IS
Mid Level Fail\
s To Off OR Level Switch Failures IS Mid and High Level Fail
To Off OR L\
evel Switch Failures IS Mid and Low Level Fail To Off OR
Level Switch Fa\
ilures IS All Fail Off)) THEN
            BEGIN
              mid level sensor OF surge bin := FALSE
              fill color OF ML sensor := 255,0,0
            END
          ELSE
```

```
            BEGIN
              mid level sensor OF surge bin := TRUE
              fill color OF ML sensor := 0,255,0
            END
          IF (use bin level AND (Level Switch Failures IS
Mid and Low Le\
vel Fail To Off OR Level Switch Failures IS All Fail Off))
THEN
            BEGIN
              low level sensor OF surge bin := FALSE
              fill color OF LL sensor := 255,0,0
            END
          ELSE
            BEGIN
              low level sensor OF surge bin := TRUE
              fill color OF LL sensor := 0,255,0
            END
          IF level OF surge bin >= 15 THEN
            BEGIN
              IF (use bin level AND (Level Switch Failures
IS Mid and Hi\
gh Level Fail To Off OR Level Switch Failures IS All Fail
Off)) THEN
                BEGIN
                  high level sensor OF surge bin := FALSE
                  fill color OF HL sensor := 255,0,0
                END
              ELSE
                BEGIN
                  high level sensor OF surge bin := TRUE
                  fill color OF HL sensor := 0,255,0
                END
            END
          ELSE
            BEGIN
              IF (use bin level AND (Level Switch Failures
IS Mid and Hi\
gh Level Fail To On OR Level Switch Failures IS All Fail
On)) THEN
                BEGIN
                  high level sensor OF surge bin := TRUE
                  fill color OF HL sensor := 0,255,0
                END
              ELSE
                BEGIN
                  high level sensor OF surge bin := FALSE
                  fill color OF HL sensor := 255,0,0
                END
            END
        END
```

```
      ELSE
        BEGIN
          IF (use bin level AND (Level Switch Failures IS
Mid and High L\
evel Fail To On OR Level Switch Failures IS All Fail On))
THEN
            BEGIN
              high level sensor OF surge bin := TRUE
              fill color OF HL sensor := 0,255,0
            END
          ELSE
            BEGIN
              high level sensor OF surge bin := FALSE
              fill color OF HL sensor := 255,0,0
            END
          IF (use bin level AND (Level Switch Failures IS
Mid Level Fail\
s To On OR Level Switch Failures IS Mid and High Level Fail
To On OR Lev\
el Switch Failures IS Mid and Low Level Fail To On OR Level
Switch Failu\
res IS All Fail On)) THEN
            BEGIN
              mid level sensor OF surge bin := TRUE
              fill color OF ML sensor := 0,255,0
            END
          ELSE
            BEGIN
              mid level sensor OF surge bin := FALSE
              fill color OF ML sensor := 255,0,0
            END
          IF level OF surge bin >= 5 THEN
            BEGIN
              IF (use bin level AND (Level Switch Failures
IS Mid and Lo\
w Level Fail To Off OR Level Switch Failures IS All Fail
Off)) THEN
                BEGIN
                  low level sensor OF surge bin := FALSE
                  fill color OF LL sensor := 255,0,0
                END
              ELSE
                BEGIN
                  low level sensor OF surge bin := TRUE
                  fill color OF LL sensor := 0,255,0
                END
            END
          ELSE
            BEGIN
```

```
              IF (use bin level AND (Level Switch Failures
IS Mid and Lo\
w Level Fail To On OR Level Switch Failures IS All Fail On))
THEN
                BEGIN
                  low level sensor OF surge bin := TRUE
                  fill color OF LL sensor := 0,255,0
                END
              ELSE
                BEGIN
                  low level sensor OF surge bin := FALSE
                  fill color OF LL sensor := 255,0,0
                END
            END
        END
      IF (use other AND Other Failures IS Discharge Screw
Plugged) THEN
        amps OF dryer discharge screw := 35
      ELSE
        amps OF dryer discharge screw := 0.06 * SQR( level
OF surge bin)\
 + 3
      IF (interval number = 1 OR NOT (use other AND Other
Failures IS Da\
mper Stuck)) THEN
        damper temp change OF the dryer := 220 - (-2 *
moisture OF wet s\
awdust + fan rate[ fan index OF the dryer] OF the dryer)
      IF damper temp change OF the dryer > 5 THEN
        damper temp change OF the dryer := 5
      IF damper temp change OF the dryer < -5 THEN
        damper temp change OF the dryer := -5
      IF (level OF surge bin = 0) OR (use other AND (Other
Failures IS D\
ischarge Screw Plugged OR (Other Failures IS Dryer Drum Fire
AND interva\
l number >= 3) OR (Other Failures IS Drag Chain Conveyer
Stuck AND level\
 OF surge bin = 0))) THEN
        BEGIN
          damper position OF the dryer := 0
          IF fan amps OF the dryer >= 12 THEN
            fan amps OF the dryer := fan amps OF the dryer -
6
          IF (use other AND Other Failures IS Dryer Drum
Fire) THEN
            BEGIN
              IF temperature OF the dryer < 602 THEN
                temperature OF the dryer := temperature OF
the dryer * 1\
```

```
.35
            END
          ELSE
            BEGIN
              IF temperature OF the dryer < 250 THEN
                temperature OF the dryer := temperature OF
the dryer * 1\
.02
            END
          IF interval number = 1 THEN
            moisture OF dry sawdust := 8
          IF moisture OF dry sawdust < 2.5 THEN
            moisture OF dry sawdust := 0 CF -2
          ELSE
            moisture OF dry sawdust := moisture OF dry
sawdust - 1.21
        END
      ELSE
        BEGIN
          fan amps OF the dryer := 6 * fan index OF the
dryer
          damper position OF the dryer := 10 * damper temp
change OF the\
 dryer + 50
          temperature OF the dryer := -2 * moisture OF wet
sawdust + fan\
 rate[ fan index OF the dryer] OF the dryer + damper temp
change OF the \
dryer
          moisture OF dry sawdust := 1212.17488 * EXP( -
0.022314 * tempe\
rature OF the dryer)
        END
      IF (use other AND Other Failures IS Drag Chain
Conveyer Stuck AND \
interval number > 2 AND on OF drag chain conveyor = 1) THEN
        BEGIN
          amps OF drag chain conveyor := 30
          on OF drag chain conveyor := 0
        END
      IF use control THEN
        BEGIN
          action OF file 2 IS open old := TRUE
          action OF bridge file IS open new := TRUE
          write line OF file 2 := TO STRING( amps OF dryer
discharge scr\
ew)
          write line OF file 2 := TO STRING( high level
sensor OF surge \
bin)
```

```
          write line OF file 2 := TO STRING( mid level
sensor OF surge b\
in)
          write line OF file 2 := TO STRING( low level
sensor OF surge b\
in)
          write line OF file 2 := TO STRING( temperature OF
the dryer)
          write line OF file 2 := TO STRING( fan index OF
the dryer)
          write line OF file 2 := TO STRING( damper position
OF the drye\
r)
          write line OF file 2 := TO STRING( moisture OF dry
sawdust)
          write line OF file 2 := TO STRING( amps OF drag
chain conveyor\
)
          write line OF bridge file := TO STRING( amps OF
dryer dischar\
e screw)
          write line OF bridge file := TO STRING( high level
sensor OF s\
urge bin)
          write line OF bridge file := TO STRING( mid level
sensor OF su\
rge bin)
          write line OF bridge file := TO STRING( low level
sensor OF su\
rge bin)
          write line OF bridge file := TO STRING(
temperature OF the dry\
er)
          write line OF bridge file := TO STRING( fan index
OF the dryer\
)
          write line OF bridge file := TO STRING( damper
position OF the\
 dryer)
          write line OF bridge file := TO STRING( moisture
OF dry sawdus\
t)
          write line OF bridge file := TO STRING( amps OF
drag chain con\
veyor)
          action OF bridge file IS close := TRUE
          action OF file 2 IS close := TRUE
          ACTIVATE  "IPU, EXTERN, ANALYZE.PIF"
        END
      ELSE
```

```
        BEGIN
          IF amps OF drag chain conveyor <> 30 THEN
            BEGIN
              IF mid level sensor OF surge bin THEN
                BEGIN
                  amps OF drag chain conveyor := 0
                  on OF drag chain conveyor := 0
                END
              ELSE
                BEGIN
                  amps OF drag chain conveyor := 20
                  on OF drag chain conveyor := 1
                END
            END
        END
    END
ATTRIBUTE Level Switch Failures COMPOUND
  Mid Level Fails To On,
  Mid Level Fails To Off,
  Mid and High Level Fail To On,
  Mid and High Level Fail To Off,
  Mid and Low Level Fail To On,
  Mid and Low Level Fail To Off,
  All Fail On,
  All Fail Off
  DEFAULT Mid Level Fails To On
ATTRIBUTE system shutdown SIMPLE
  WHEN CHANGED
    BEGIN
      sample period.active := FALSE
      active OF shutdown timer := TRUE
      start OF shutdown timer := NOW
    END
ATTRIBUTE start shutdown SIMPLE
  WHEN CHANGED
    BEGIN
      amps OF drag chain conveyor := 0
      amps OF dryer discharge screw := 0
      IF flow rate OF the dryer > 2 THEN
        flow rate OF the dryer := flow rate OF the dryer /
1.5
      ELSE
        BEGIN
          flow rate OF the dryer := 0
          moisture OF dry sawdust := 0 CF -2
        END
      IF temperature OF the dryer > 2 THEN
        temperature OF the dryer := temperature OF the dryer
/ 1.1
      ELSE
```

```
        temperature OF the dryer := 0
      damper position OF the dryer := 50
      fan amps OF the dryer := 0
    END
ATTRIBUTE select level switch failures SIMPLE
  WHEN CHANGED
    BEGIN
      IF moisture OF wet sawdust > 70 THEN
        moisture OF wet sawdust := 47
      fill color OF normal light := 192,192,192
      fill color OF bin faults light := 0,0,0
      fill color OF other light := 192,192,192
      visible OF control window := FALSE
      title OF control window := "SURGE BIN FAULT
SIMULATIONS"
      output OF control window := bin faults screen
      visible OF control window := TRUE
      use bin level := TRUE
      use normal := FALSE
      use other := FALSE
    END
ATTRIBUTE select normal operations SIMPLE
  WHEN CHANGED
    BEGIN
      IF moisture OF wet sawdust > 70 THEN
        moisture OF wet sawdust := 47
      fill color OF normal light := 0,0,0
      fill color OF bin faults light := 192,192,192
      fill color OF other light := 192,192,192
      use other := FALSE
      use bin level := FALSE
      use normal := TRUE
    END
ATTRIBUTE select other failures SIMPLE
  WHEN CHANGED
    BEGIN
      fill color OF normal light := 192,192,192
      fill color OF bin faults light := 192,192,192
      fill color OF other light := 0,0,0
      use normal := FALSE
      use bin level := FALSE
      use other := TRUE
      visible OF control window := FALSE
      title OF control window := "SYSTEM FAULT SIMULATIONS"
      output OF control window := other faults display
      visible OF control window := TRUE
    END
ATTRIBUTE use normal SIMPLE
ATTRIBUTE use bin level SIMPLE
ATTRIBUTE use other SIMPLE
```

```
ATTRIBUTE interval number NUMERIC
ATTRIBUTE true SIMPLE
ATTRIBUTE Other Failures COMPOUND
  Drag Chain Conveyer Stuck,
  Discharge Screw Plugged,
  Damper Stuck,
  Enable High Moisture Range,
  Dryer Drum Fire
  DEFAULT Drag Chain Conveyer Stuck
ATTRIBUTE select control SIMPLE
  WHEN CHANGED
    BEGIN
      use control := NOT use control
      IF text OF control or no contro = "Network" THEN
        BEGIN
          text OF control or no contro := "No Net"
          text OF report field := "
Control diagnostics module not connected."
          period OF sample period := 0 00:00:04
        END
      ELSE
        BEGIN
          text OF control or no contro := "Network"
          text OF report field := "
Neural Network controller on-line."
          period OF sample period := 0 00:00:10
        END
    END
ATTRIBUTE use control SIMPLE
ATTRIBUTE select initial conditions SIMPLE
  WHEN CHANGED
    BEGIN
      visible OF control window := FALSE
      title OF control window := "INITIAL CONDITIONS"
      label OF drag chain on off := "Drag Chain"
      output OF control window := initial conditons
      visible OF control window := TRUE
    END
ATTRIBUTE init drag chain SIMPLE
  WHEN CHANGED
    BEGIN
      IF on OF drag chain conveyor = 1 THEN
        BEGIN
          label OF drag chain on off := "Drag Chain OFF"
          on OF drag chain conveyor := 0
          amps OF drag chain conveyor := 0
        END
      ELSE
        BEGIN
          label OF drag chain on off := "Drag Chain ON"
```

```
          on OF drag chain conveyor := 1
          amps OF drag chain conveyor := 20
        END
    END
ATTRIBUTE implement sim control option SIMPLE
  WHEN CHANGED
    BEGIN
      IF (use control AND interval number > 1) THEN
        BEGIN
          action OF bridge file IS open old := TRUE
          read line OF bridge file := TRUE
          control code := TO NUMERIC( current line OF bridge
file)
          IF control code = 1 THEN
            BEGIN
              read line OF bridge file := TRUE
              system fault := current line OF bridge file
              text OF aux reports box := CONCAT( "**
CRITICAL SYSTEM FAU\
LT DETECTED **

", system fault)
              text OF aux reports box := CONCAT( text OF aux
reports box\
, "

Implementing Shutdown")
              action OF bridge file IS close := TRUE
              shutdown := TRUE
              system shutdown := NOT (system shutdown)
            END
          ELSE
            BEGIN
              IF control code = 2 THEN
                text OF aux reports box := "System OK"
              ELSE
                BEGIN
                  read line OF bridge file := TRUE
                  system fault := current line OF bridge
file
                  text OF aux reports box := CONCAT( "**
TOLERABLE SYSTE\
M FAULT DETECTED **

 ", system fault)
                END
              read line OF bridge file := TRUE
              on OF drag chain conveyor := TO NUMERIC(
current line OF b\
ridge file)
```

```
                IF on OF drag chain conveyor = 1 THEN
                  amps OF drag chain conveyor := 20
                ELSE
                  amps OF drag chain conveyor := 0
                read line OF bridge file := TRUE
                fan index OF the dryer := TO NUMERIC( current
line OF brid\
ge file)
                action OF bridge file IS close := TRUE
                compute values := NOT (compute values)
              END
          END
        ELSE
          compute values := NOT (compute values)
    END
ATTRIBUTE system fault STRING
ATTRIBUTE control code NUMERIC
ATTRIBUTE shutdown SIMPLE
ATTRIBUTE pause SIMPLE
  WHEN CHANGED
    BEGIN
      IF text OF pause box = "Simulation
Paused" THEN
        BEGIN
          IF shutdown THEN
            active OF shutdown timer := TRUE
          ELSE
            active OF sample period := TRUE
          text OF pause box := ""
        END
      ELSE
        BEGIN
          text OF pause box := "Simulation
Paused"
          active OF shutdown timer := FALSE
          active OF sample period := FALSE
        END
    END
ATTRIBUTE check bin SIMPLE
  WHEN CHANGED
    BEGIN
      level OF surge bin := TRUNC( level OF surge bin)
      IF level OF surge bin < 3 THEN
        level OF surge bin := 3
      IF level OF surge bin >= 19 THEN
        level OF surge bin := 19
      location OF bin level := SETRECT( 171, 216 - (level OF
surge bin /\
 20 * 97), 183, 216)
    END
```

```
ATTRIBUTE exp NUMERIC

INSTANCE the domain ISA domain
  WITH stop := FALSE
  WITH Level Switch Failures IS Mid Level Fails To On
  WITH system shutdown := FALSE
  WITH start shutdown := FALSE
  WITH select level switch failures := FALSE
  WITH select normal operations := TRUE
  WITH select other failures := FALSE
  WITH use normal := TRUE
  WITH use bin level := FALSE
  WITH use other := FALSE
  WITH interval number := 0
  WITH Other Failures IS Drag Chain Conveyer Stuck
  WITH select control := FALSE
  WITH use control := FALSE
  WITH init drag chain := FALSE
  WITH shutdown := FALSE

INSTANCE the application ISA application
  WITH unknowns fail := TRUE
  WITH threshold := 50
  WITH title display := dryer system overview
  WITH ignore breakpoints := FALSE
  WITH reasoning on := FALSE
  WITH numeric precision := 8
  WITH simple query text := "Is it true that:
  *
is
  *"
  WITH numeric query text := "What is(are):
  *
of
  *"
  WITH string query text := "What is(are):
  *
of
  *"
  WITH time query text := "What is(are):
  *
of
  *"
  WITH interval query text := "What is(are):
  *
of
  *"
  WITH compound query text := "What is(are):
  *
of
```

```
    *"
    WITH multicompound query text := "What is(are):
    *
  of
    *"
    WITH demon strategy IS fire first
    WITH visible file menu := FALSE

  INSTANCE dryer system overview ISA display
    WITH wait := FALSE
    WITH delay changes := FALSE
    WITH items [1 ] := report field
    WITH items [2 ] := dryer system
    WITH items [3 ] := dos window field
    WITH items [4 ] := process reports
    WITH items [5 ] := simulator controls
    WITH items [6 ] := start button
    WITH items [7 ] := dryer temp
    WITH items [8 ] := ID fan speed
    WITH items [9 ] := damper positoin
    WITH items [10 ] := dryer flow rate
    WITH items [11 ] := bin level
    WITH items [12 ] := wet sawdust moisture
    WITH items [13 ] := drag chain amps
    WITH items [14 ] := discharge screw amps
    WITH items [15 ] := dry sawdust moisture
    WITH items [16 ] := plus moisture
    WITH items [17 ] := minus moisture
    WITH items [18 ] := valuebox 9
    WITH items [19 ] := control panel writing input sawdust
    WITH items [20 ] := HL sensor
    WITH items [21 ] := ML sensor
    WITH items [22 ] := LL sensor
    WITH items [23 ] := stop button
    WITH items [24 ] := exit button
    WITH items [25 ] := normal ops
    WITH items [26 ] := bin faults
    WITH items [27 ] := system faults
    WITH items [28 ] := normal light
    WITH items [29 ] := bin faults light
    WITH items [30 ] := other light
    WITH items [31 ] := normal operations
    WITH items [32 ] := level faults
    WITH items [33 ] := other faults
    WITH items [34 ] := timestep
    WITH items [35 ] := time step
    WITH items [36 ] := init cond
    WITH items [37 ] := control
    WITH items [38 ] := control or no contro
    WITH items [39 ] := initial
```

```
  WITH items [40 ] := conditions
  WITH items [41 ] := aux reports box
  WITH items [42 ] := pause button
  WITH items [43 ] := pause box

INSTANCE bin faults screen ISA display
  WITH wait := FALSE
  WITH delay changes := FALSE
  WITH items [1 ] := UNDETERMINED
  WITH items [2 ] := radiobutton group 2

INSTANCE other faults display ISA display
  WITH wait := FALSE
  WITH delay changes := FALSE
  WITH items [1 ] := radiobutton group 3

INSTANCE initial conditons ISA display
  WITH wait := FALSE
  WITH delay changes := FALSE
  WITH items [1 ] := UNDETERMINED
  WITH items [2 ] := bin level up
  WITH items [3 ] := bin level down
  WITH items [4 ] := fan index down
  WITH items [5 ] := fan index up
  WITH items [6 ] := init index
  WITH items [7 ] := init bin level
  WITH items [8 ] := init surge level
  WITH items [9 ] := initial index
  WITH items [10 ] := drag chain on off

INSTANCE bridge file ISA file
  WITH filename := "data.txt"
  WITH shared := TRUE

INSTANCE file 2 ISA file
  WITH filename := "test.txt"
  WITH shared := TRUE

INSTANCE dryer system ISA picturebox
  WITH location := 5,5,625,495
  WITH clipped := TRUE
  WITH frame := TRUE
  WITH filename := "C:\\PROJECT\\DRYER.BMP"

INSTANCE simulator controls ISA picturebox
  WITH location := 5,500,625,738
  WITH clipped := TRUE
  WITH filename := "C:\\PROJECT\\CTLPANEL.BMP"

INSTANCE plus moisture ISA pushbutton
```

```
  WITH location := 330,682,350,702
  WITH label := "+"
  WITH attribute attachment := moisture up OF wet sawdust

INSTANCE minus moisture ISA pushbutton
  WITH location := 350,682,370,702
  WITH label := "-"
  WITH attribute attachment := moisture down OF wet sawdust

INSTANCE bin level up ISA pushbutton
  WITH location := 179,65,199,85
  WITH label := "+"
  WITH attribute attachment := init bin level up OF surge
bin

INSTANCE bin level down ISA pushbutton
  WITH location := 199,65,219,85
  WITH label := "-"
  WITH attribute attachment := init bin level down OF surge
bin

INSTANCE fan index down ISA pushbutton
  WITH location := 55,65,75,85
  WITH label := "-"
  WITH attribute attachment := init fan index down OF the
dryer

INSTANCE fan index up ISA pushbutton
  WITH location := 35,65,55,85
  WITH label := "+"
  WITH attribute attachment := init fan index up OF the
dryer

INSTANCE drag chain on off ISA pushbutton
  WITH location := 70,140,184,170
  WITH label := "Drag Chain OFF"
  WITH attribute attachment := init drag chain

INSTANCE radiobutton group 2 ISA radiobutton group
  WITH location := 0,0,250,180
  WITH pen color := 0,0,0
  WITH fill color := 255,255,255
  WITH frame := FALSE
  WITH group label := "Level Switch Failures"
  WITH show default := FALSE
  WITH show current := TRUE
  WITH attachment := Level Switch Failures OF the domain

INSTANCE radiobutton group 3 ISA radiobutton group
  WITH location := 0,0,250,180
```

```
  WITH frame := FALSE
  WITH group label := "Other Failures"
  WITH show default := FALSE
  WITH show current := TRUE
  WITH attachment := Other Failures OF the domain

INSTANCE dos window field ISA textbox
  WITH location := 630,500,1019,738
  WITH pen color := 0,0,0
  WITH fill color := 0,0,0
  WITH justify IS left
  WITH font := "System"
  WITH frame := TRUE
  WITH text := ""

INSTANCE report field ISA textbox
  WITH location := 630,40,1019,495
  WITH justify IS center
  WITH font := "System"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 10
  WITH frame := TRUE
  WITH text := "
Control diagnostics module not connected."

INSTANCE process reports ISA textbox
  WITH location := 630,5,1019,40
  WITH pen color := 255,255,255
  WITH fill color := 0,0,0
  WITH justify IS center
  WITH font := "MS Sans Serif"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 14
  WITH frame := TRUE
  WITH text := "Process Reports"

INSTANCE bin level ISA textbox
  WITH location := 171,153,183,216
  WITH pen color := 128,64,0
  WITH fill color := 255,255,255
  WITH justify IS left
  WITH font := "System"
  WITH text := ""

INSTANCE control panel writing input sawdust ISA textbox
  WITH location := 320,630,382,645
```

```
  WITH pen color := 0,0,0
  WITH fill color := 192,192,192
  WITH justify IS left
  WITH font := "Small Fonts"
  WITH font style IS bold CF FALSE, italic CF FALSE,
underline CF FALSE,\
 strikeout CF FALSE
  WITH font size := 6
  WITH text := "Wet Sawdust"

INSTANCE HL sensor ISA textbox
  WITH location := 238,137,248,147
  WITH pen color := 255,0,0
  WITH fill color := 0,0,0
  WITH justify IS left
  WITH font := "System"
  WITH frame := TRUE
  WITH text := ""

INSTANCE ML sensor ISA textbox
  WITH location := 238,163,248,173
  WITH pen color := 255,0,0
  WITH fill color := 0,0,0
  WITH justify IS left
  WITH font := "System"
  WITH frame := TRUE
  WITH text := ""

INSTANCE LL sensor ISA textbox
  WITH location := 238,190,248,200
  WITH pen color := 255,0,0
  WITH fill color := 0,0,0
  WITH justify IS left
  WITH font := "System"
  WITH frame := TRUE
  WITH text := ""

INSTANCE normal ops ISA textbox
  WITH location := 389,538,453,553
  WITH pen color := 0,0,0
  WITH fill color := 192,192,192
  WITH justify IS left
  WITH font := "Small Fonts"
  WITH font style IS bold CF FALSE, italic CF FALSE,
underline CF FALSE,\
 strikeout CF FALSE
  WITH font size := 6
  WITH scroll := FALSE
  WITH text := "Normal Ops"
```

```
INSTANCE bin faults ISA textbox
  WITH location := 389,563,453,578
  WITH pen color := 0,0,0
  WITH fill color := 192,192,192
  WITH justify IS left
  WITH font := "Small Fonts"
  WITH font style IS bold CF FALSE, italic CF FALSE,
underline CF FALSE,\
 strikeout CF FALSE
  WITH font size := 6
  WITH text := "Bin Faults"

INSTANCE system faults ISA textbox
  WITH location := 389,588,453,603
  WITH pen color := 0,0,0
  WITH fill color := 192,192,192
  WITH justify IS left
  WITH font := "Small Fonts"
  WITH font style IS bold CF FALSE, italic CF FALSE,
underline CF FALSE,\
 strikeout CF FALSE
  WITH font size := 6
  WITH frame := FALSE
  WITH text := "Other Faults"

INSTANCE normal light ISA textbox
  WITH location := 340,540,350,550
  WITH pen color := 0,0,0
  WITH fill color := 0,0,0
  WITH justify IS left
  WITH font := "System"
  WITH frame := TRUE
  WITH text := ""

INSTANCE bin faults light ISA textbox
  WITH location := 340,565,350,575
  WITH pen color := 0,0,0
  WITH fill color := 192,192,192
  WITH justify IS left
  WITH font := "System"
  WITH frame := TRUE
  WITH text := ""

INSTANCE other light ISA textbox
  WITH location := 340,590,350,600
  WITH pen color := 0,0,0
  WITH fill color := 192,192,192
  WITH justify IS left
  WITH font := "System"
  WITH frame := TRUE
```

```
  WITH text := ""

INSTANCE timestep ISA textbox
  WITH location := 420,620,599,640
  WITH pen color := 0,0,0
  WITH fill color := 192,192,192
  WITH justify IS left
  WITH font := "MS Sans Serif"
  WITH font style IS bold CF FALSE, italic CF FALSE,
underline CF FALSE,\
 strikeout CF FALSE
  WITH font size := 7
  WITH text := "Time-Step:              @ 5 min / interval"

INSTANCE control or no contro ISA textbox
  WITH location := 525,582,575,595
  WITH pen color := 0,0,0
  WITH fill color := 192,192,192
  WITH justify IS center
  WITH font := "Small Fonts"
  WITH font style IS bold CF FALSE, italic CF FALSE,
underline CF FALSE,\
 strikeout CF FALSE
  WITH font size := 6
  WITH text := "No Net"

INSTANCE initial ISA textbox
  WITH location := 475,582,520,594
  WITH pen color := 0,0,0
  WITH fill color := 192,192,192
  WITH justify IS center
  WITH font := "Small Fonts"
  WITH font style IS bold CF FALSE, italic CF FALSE,
underline CF FALSE,\
 strikeout CF FALSE
  WITH font size := 6
  WITH text := "Initial"

INSTANCE conditions ISA textbox
  WITH location := 473,595,528,607
  WITH pen color := 0,0,0
  WITH fill color := 192,192,192
  WITH justify IS left
  WITH font := "Small Fonts"
  WITH font style IS bold CF FALSE, italic CF FALSE,
underline CF FALSE,\
 strikeout CF FALSE
  WITH font size := 6
  WITH text := "Conditions"
```

```
INSTANCE init surge level ISA textbox
  WITH location := 155,90,235,105
  WITH pen color := 0,0,0
  WITH fill color := 255,255,255
  WITH justify IS center
  WITH font := "Small Fonts"
  WITH font style IS bold CF FALSE, italic CF FALSE,
underline CF FALSE,\
 strikeout CF FALSE
  WITH font size := 6
  WITH text := "Surge Bin Level"

INSTANCE initial index ISA textbox
  WITH location := 20,90,90,120
  WITH pen color := 0,0,0
  WITH fill color := 255,255,255
  WITH justify IS center
  WITH font := "Small Fonts"
  WITH font style IS bold CF FALSE, italic CF FALSE,
underline CF FALSE,\
 strikeout CF FALSE
  WITH font size := 6
  WITH text := "Fan Speed
Setting"

INSTANCE aux reports box ISA textbox
  WITH location := 635,135,1014,320
  WITH justify IS center
  WITH font := "System"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 10
  WITH frame := FALSE
  WITH text := ""

INSTANCE pause box ISA textbox
  WITH location := 65,555,270,685
  WITH pen color := 0,0,0
  WITH fill color := 192,192,192
  WITH justify IS center
  WITH font := "MS Sans Serif"
  WITH font style IS bold CF FALSE, italic CF FALSE,
underline CF FALSE,\
 strikeout CF FALSE
  WITH font size := 18
  WITH text := ""

INSTANCE sample period ISA timer
  WITH period := 0 00:00:04.000
```

```
INSTANCE shutdown timer ISA timer
  WITH period := 0 00:00:03.000

INSTANCE dryer temp ISA valuebox
  WITH location := 469,281,510,301
  WITH pen color := 255,0,0
  WITH fill color := 0,0,0
  WITH justify IS center
  WITH font := "MS Sans Serif"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 8
  WITH frame := FALSE
  WITH clipped := TRUE
  WITH format := "###.#"
  WITH attachment := temperature OF the dryer

INSTANCE ID fan speed ISA valuebox
  WITH location := 472,180,513,200
  WITH pen color := 255,0,0
  WITH fill color := 0,0,0
  WITH justify IS center
  WITH font := "MS Sans Serif"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 8
  WITH frame := FALSE
  WITH clipped := FALSE
  WITH attachment := fan amps OF the dryer

INSTANCE damper positoin ISA valuebox
  WITH location := 330,217,371,237
  WITH pen color := 255,0,0
  WITH fill color := 0,0,0
  WITH justify IS center
  WITH font := "MS Sans Serif"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 8
  WITH frame := FALSE
  WITH clipped := TRUE
  WITH attachment := damper position OF the dryer

INSTANCE dryer flow rate ISA valuebox
  WITH location := 212,348,252,368
  WITH pen color := 255,0,0
```

```
  WITH fill color := 0,0,0
  WITH justify IS center
  WITH font := "MS Sans Serif"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 8
  WITH frame := FALSE
  WITH clipped := TRUE
  WITH format := "##.0"
  WITH attachment := flow rate OF the dryer

INSTANCE wet sawdust moisture ISA valuebox
  WITH location := 30,170,73,185
  WITH pen color := 0,0,255
  WITH fill color := 255,255,255
  WITH justify IS center
  WITH font := "MS Sans Serif"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 8
  WITH frame := FALSE
  WITH clipped := FALSE
  WITH attachment := moisture OF wet sawdust

INSTANCE drag chain amps ISA valuebox
  WITH location := 50,48,95,65
  WITH pen color := 0,0,255
  WITH fill color := 255,255,255
  WITH justify IS center
  WITH font := "MS Sans Serif"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 8
  WITH frame := FALSE
  WITH clipped := FALSE
  WITH attachment := amps OF drag chain conveyor

INSTANCE discharge screw amps ISA valuebox
  WITH location := 295,134,340,151
  WITH pen color := 0,0,255
  WITH fill color := 255,255,255
  WITH justify IS center
  WITH font := "MS Sans Serif"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 8
```

```
  WITH frame := FALSE
  WITH clipped := FALSE
  WITH format := "##.0"
  WITH attachment := amps OF dryer discharge screw

INSTANCE dry sawdust moisture ISA valuebox
  WITH location := 485,440,530,457
  WITH pen color := 0,0,255
  WITH fill color := 255,255,255
  WITH justify IS center
  WITH font := "MS Sans Serif"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 8
  WITH frame := FALSE
  WITH clipped := FALSE
  WITH format := "##.0"
  WITH attachment := moisture OF dry sawdust

INSTANCE valuebox 9 ISA valuebox
  WITH location := 330,650,369,675
  WITH justify IS center
  WITH font := "System"
  WITH frame := TRUE
  WITH clipped := TRUE
  WITH attachment := moisture OF wet sawdust

INSTANCE time step ISA valuebox
  WITH location := 475,618,505,640
  WITH pen color := 0,0,0
  WITH fill color := 192,192,192
  WITH justify IS center
  WITH font := "MS Sans Serif"
  WITH font style IS bold, italic CF FALSE, underline CF
FALSE, strikeou\
t CF FALSE
  WITH font size := 8
  WITH frame := FALSE
  WITH clipped := TRUE
  WITH attachment := interval number OF the domain

INSTANCE init index ISA valuebox
  WITH location := 35,35,75,60
  WITH pen color := 255,255,255
  WITH fill color := 0,0,0
  WITH justify IS center
  WITH font := "System"
  WITH frame := TRUE
  WITH clipped := TRUE
```

```
  WITH attachment := fan index OF the dryer

INSTANCE init bin level ISA valuebox
  WITH location := 179,35,219,60
  WITH pen color := 255,255,255
  WITH fill color := 0,0,0
  WITH justify IS center
  WITH font := "System"
  WITH frame := TRUE
  WITH clipped := TRUE
  WITH format := "##"
  WITH attachment := level OF surge bin

INSTANCE main window ISA window
  WITH location := -1,-1,-1,-1
  WITH full screen := TRUE
  WITH style IS moveable CF FALSE, sizeable CF FALSE,
closeable CF FALSE
  WITH title := "DRYER SIM"
  WITH visible := TRUE
  WITH visible OK button := FALSE

INSTANCE control window ISA window
  WITH location := 36,540,296,751
  WITH full screen := FALSE
  WITH style IS moveable CF FALSE, sizeable CF FALSE,
closeable CF FALSE
  WITH title := "SURGE BIN FAULT SIMULATIONS"
  WITH visible := FALSE
  WITH output := bin faults screen
  WITH visible OK button := FALSE

DEMON for starting simulation
IF start
THEN sample period.active := TRUE
AND sample period.start := NOW
AND text OF aux reports box := ""
AND enabled OF start button := FALSE
AND enabled OF normal operations := FALSE
AND enabled OF level faults := FALSE
AND enabled OF other faults := FALSE
AND enabled OF init cond := FALSE
AND enabled OF control := FALSE
AND fill color OF HL sensor := 0,0,0
AND fill color OF ML sensor := 0,0,0
AND fill color OF LL sensor := 0,0,0
AND enabled OF stop button := TRUE
AND enabled OF pause button := TRUE
AND flow rate OF the dryer := 0 CF -2
AND damper position OF the dryer := 0 CF -2
```

```
AND temperature OF the dryer := 220
AND amps OF dryer discharge screw := 0 CF -2
AND moisture OF dry sawdust := 0 CF -2
AND interval number := 0

DEMON for stopping simulation
IF stop
THEN sample period.active := FALSE
AND active OF shutdown timer := FALSE
AND shutdown := FALSE
AND text OF pause box := ""
AND enabled OF start button := TRUE
AND enabled OF normal operations := TRUE
AND enabled OF level faults := TRUE
AND enabled OF other faults := TRUE
AND enabled OF init cond := TRUE
AND enabled OF control := TRUE
AND enabled OF stop button := FALSE
AND enabled OF pause button := FALSE
AND amps OF drag chain conveyor := 0
AND on OF drag chain conveyor := 0
AND check bin

DEMON for program termination
IF exit
THEN CHAIN "simexit.knb"

DEMON for starting sim computations
IF tripped OF sample period
THEN interval number := interval number + 1
AND implement sim control option

DEMON for simulated shutdown
IF tripped OF shutdown timer
THEN start shutdown

END



$VERSION25
$LOCATIONS ARE PIXELS


INSTANCE the application ISA application
  WITH unknowns fail := TRUE
  WITH threshold := 50
  WITH title display := exit display
  WITH ignore breakpoints := FALSE
  WITH reasoning on := FALSE
```

```
  WITH numeric precision := 8
  WITH simple query text := "Is it true that:
  *
is
  *"
  WITH numeric query text := "What is(are):
  *
of
  *"
  WITH string query text := "What is(are):
  *
of
  *"
  WITH time query text := "What is(are):
  *
of
  *"
  WITH interval query text := "What is(are):
  *
of
  *"
  WITH compound query text := "What is(are):
  *
of
  *"
  WITH multicompound query text := "What is(are):
  *
of
  *"
  WITH demon strategy IS fire first
  WITH visible file menu := TRUE

INSTANCE exit display ISA display
  WITH wait := TRUE
  WITH delay changes := FALSE
  WITH items [1 ] := textbox 9

INSTANCE textbox 9 ISA textbox
  WITH location := 165,45,610,335
  WITH pen color := 255,255,255
  WITH fill color := 0,0,255
  WITH justify IS center
  WITH font := "MS Sans Serif"
  WITH font style IS bold CF FALSE, italic CF FALSE,
underline CF FALSE,\
 strikeout CF FALSE
  WITH font size := 18
  WITH frame := TRUE
  WITH text := "
```

```
Simulation Terminated"

INSTANCE main window ISA window
  WITH location := 144,163,924,613
  WITH style IS moveable, sizeable, closeable
  WITH title := "DRYER SIM"
  WITH visible := TRUE
  WITH visible OK button := TRUE

END
```

APPENDIX C

NEURAL NETWORK CONTROLLER SOURCE CODE

```
program ANALYZE;

(*****************************************************************)
(*                          ANALYZE                              *)
(*****************************************************************)
(*  By: Doulas M. Rogers                                         *)
(*  Last update: 11 Feb 94                                       *)
(*                                                               *)
(*   This program is an expert network controller module         *)
(* used in controlling the DRYER-SIM process simulator.          *)
(* The program receives information from DRYER-SIM via a         *)
(* text file (DATA.TXT).  These inputs are used by the           *)
(* neural networks contained in this program to produce          *)
(* system state outputs that control the DRYERSIM.  The          *)
(* outputs are placed back into DATA.TXT where DRYER-SIM         *)
(* program expect to find them at the proper sample time.        *)
(* This program also utilizes two other data files:              *)
(* WTS1.TXT and WTS2.TXT.  These files contain the               *)
(*  connection weight matricies for the neural nets as           *)
(* determined by the training program (TRAINER.PAS).  In         *)
(* this way, the neural nets may be retrained using              *)
(* TRAINER.PAS, and no direct modification to this               *)
(*  program is required                                          *)
(*                                                               *)
(*****************************************************************)


type
   ARRAYTYPE1 = array[1..5] of real;
   ARRAYTYPE2 = array[1..10] of real;
   ARRAYTYPE3 = array[1..10] of string[40];

var
   DATA_FILE      : text;       (* interface text file       *)
   BIN_ARRAY,                   (* array of bin input data   *)
   DRY_ARRAY      : ARRAYTYPE1; (* array of dryer input data*)
   OUT1_ARRAY,                  (* bin network output array  *)
   OUT2_ARRAY     : ARRAYTYPE2; (* dryer network output      *)
   TEXT1_ARRAY,                 (* array of fault text       *)
   TEXT2_ARRAY    : ARRAYTYPE3; (* array of fault text       *)
   TEXT_DATA      : char;       (* used for non-numeric data*)
   LOOP           : integer;    (* loop control variable     *)


procedure GET_RESULTS(      IN_LAYER  : ARRAYTYPE1;
                            NUM_IN,
                            NUM_HID,
                            NUM_OUT   : integer;
                        var OUT_LAYER : ARRAYTYPE2 );
```

```
var
   LOOP1,                                          (* loop control *)
   LOOP2     : integer;                            (* loop control *)
   WTS_FILE  : text;                               (* weights file *)
   HID_LAYER : ARRAYTYPE2;                         (* hidden layer *)
   WT_IN_HID : array[1..6,  1..4] of real;(* weight array *)
   WT_HID_OUT: array[1..10, 1..6] of real;(* weight array *)


begin (* GET_RESULTS *)
   if NUM_HID = 4 then
      assign(WTS_FILE,'WEIGHT2.TXT')
   else
      assign(WTS_FILE,'WEIGHT1.TXT');

   reset(WTS_FILE);
   for LOOP1 := 1 to NUM_HID do
      for LOOP2 := 1 to NUM_IN do
         readln(WTS_FILE, WT_IN_HID[LOOP1, LOOP2]);
   for LOOP1 := 1 to NUM_OUT do
      for LOOP2 := 1 to NUM_HID do
         readln(WTS_FILE, WT_HID_OUT[LOOP1, LOOP2]);
   close(WTS_FILE);


   for LOOP1 := 1 to NUM_HID do
      begin (* LOOP1 *)
         HID_LAYER[LOOP1] := 0;
         for LOOP2 := 1 to NUM_IN do
            HID_LAYER[LOOP1] := HID_LAYER[LOOP1] +
              WT_IN_HID[LOOP1,LOOP2] * IN_LAYER[LOOP2];
         HID_LAYER[LOOP1] := 1 / ( 1 + exp(
           HID_LAYER[LOOP1]));
      end;  (* LOOP1 *)
   for LOOP1 := 1 to NUM_OUT do
      begin (* LOOP1 *)
         OUT_LAYER[LOOP1] := 0;
         for LOOP2 := 1 to NUM_HID do
            OUT_LAYER[LOOP1] := OUT_LAYER[LOOP1] +
              WT_HID_OUT[LOOP1,LOOP2] * HID_LAYER[LOOP2];
         OUT_LAYER[LOOP1] := 1 / (1 + exp(
            OUT_LAYER[LOOP1]));
      end; (* LOOP 1*)
end;(* GET_RESULTS *)


(************************ MAIN ****************************)
```

```
begin (* ANALYZE *)

   clrscr;
   writeln;
   writeln('*********************************************');
   writeln('*       DRYERSIM  ANALYSIS  MODULE          *');
   writeln('*                                           *');
   writeln('*                                           *');
   writeln('*     Processing system parameters...       *');
   writeln('*                                           *');
   writeln('*                                           *');
   write('*********************************************');

   assign(DATA_FILE,'DATA.TXT');
   reset(DATA_FILE);
   readln(DATA_FILE, BIN_ARRAY[1]);
   for LOOP := 2 to 4 do
      begin
         readln(DATA_FILE, TEXT_DATA);
         if TEXT_DATA = 'T' then
            BIN_ARRAY[LOOP] := 0.999
         else
            BIN_ARRAY[LOOP] := 0.001;
      end;
   for LOOP := 1 to 5 do
      readln(DATA_FILE, DRY_ARRAY[LOOP]);
   close(DATA_FILE);

  TEXT1_ARRAY[2]:='SURGE BIN: mid-level switch stuck on';
  TEXT1_ARRAY[3]:='SURGE BIN: mid-level switch stuck off';
  TEXT1_ARRAY[4]:='SURGE BIN: mid/high level switches on';
  TEXT1_ARRAY[5]:='SURGE BIN: mid/high level switches off';
  TEXT1_ARRAY[6]:='SURGE BIN: mid/low level switches on';
  TEXT1_ARRAY[7]:='SURGE BIN: mid/low level switches off';
  TEXT1_ARRAY[8]:='SURGE BIN: all level switches stuck on';
  TEXT1_ARRAY[9]:='SURGE BIN: all level switches stuck off';
  TEXT1_ARRAY[10]:='DRYER INFEED: discharge screw plugged';

  TEXT2_ARRAY[3]:='DRYER: temperature control damper stuck';
  TEXT2_ARRAY[4]:='DRYER: input sawdust moisture too high';
  TEXT2_ARRAY[5]:='DRYER: FIRE in the dryer drum detected';
  TEXT2_ARRAY[6]:='SURGE BIN INFEED: drag chain stuck';

   OUT2_ARRAY[8] := BIN_ARRAY[1];
   if OUT2_ARRAY[8] < 4.5 then
      BIN_ARRAY[1] := 0.001;
   if (OUT2_ARRAY[8] < 16.5) and (OUT2_ARRAY[8] >= 4.5) then
      BIN_ARRAY[1] := 0.4;
   if OUT2_ARRAY[8] >=16.5 then
      BIN_ARRAY[1] := 0.6;
```

```
if OUT2_ARRAY[8] >= 35 then
   BIN_ARRAY[1] := 0.999;

BIN_ARRAY[5] := BIN_ARRAY[1];

OUT2_ARRAY[8] := DRY_ARRAY[1];
if DRY_ARRAY[1] > 500 then
   OUT2_ARRAY[5] := 1
else
   OUT2_ARRAY[5] := 0;
if OUT2_ARRAY[8] > 225 then
   DRY_ARRAY[1] := 0.999;
if OUT2_ARRAY[8] < 215 then
   DRY_ARRAY[1] := 0.001;
if (OUT2_ARRAY[8] >= 215) and (OUT2_ARRAY[8] <= 225) then
   DRY_ARRAY[1] := 0.5;

OUT2_ARRAY[9] := DRY_ARRAY[2];
case trunc(OUT2_ARRAY[9]) of
      1 : DRY_ARRAY[2] := 0.111;
      2 : DRY_ARRAY[2] := 0.333;
      3 : DRY_ARRAY[2] := 0.555;
      4 : DRY_ARRAY[2] := 0.777;
      5 : DRY_ARRAY[2] := 0.999;
   end; (* CASE *)

OUT2_ARRAY[8] := DRY_ARRAY[3];
if OUT2_ARRAY[8] > 50 then
   DRY_ARRAY[3] := 0.999;
if OUT2_ARRAY[8] < 50 then
   DRY_ARRAY[3] := 0.001;
if OUT2_ARRAY[8] = 50 then
   DRY_ARRAY[3] := 0.5;

OUT2_ARRAY[8] := DRY_ARRAY[4];
if OUT2_ARRAY[8] > 10 then
   DRY_ARRAY[4] := 0.999;
if OUT2_ARRAY[8] < 8 then
   DRY_ARRAY[4] := 0.001;
if (OUT2_ARRAY[8] >= 8) and (OUT2_ARRAY[8] <= 10) then
   DRY_ARRAY[4] := 0.5;

GET_RESULTS(BIN_ARRAY, 4, 6, 9, OUT1_ARRAY);
GET_RESULTS(DRY_ARRAY, 4, 4, 4, OUT2_ARRAY);

OUT2_ARRAY[8] := 0;

if BIN_ARRAY[5] = 0.999 then
   OUT1_ARRAY[10] := 1
else
```

```
      OUT1_ARRAY[10] := 0;

   if DRY_ARRAY[5] >= 30 then
      OUT2_ARRAY[6] := 1
   else
      OUT2_ARRAY[6] := 0;

   for LOOP := 2 to 10 do
      begin (* FOR *)
         if OUT1_ARRAY[LOOP] > 0.5 then
            OUT2_ARRAY[8] := LOOP;
         OUT2_ARRAY[10] := 1;
      end; (* FOR *)
   for LOOP := 3 to 6 do
      begin (* FOR *)
         if OUT2_ARRAY[LOOP] > 0.5 then
            begin (* THEN *)
               OUT2_ARRAY[8] := LOOP;
               OUT2_ARRAY[10] := 2;
            end;  (* THEN *)
      end;  (* FOR *)

   if OUT2_ARRAY[8] > 0 then
      OUT2_ARRAY[7] := 3
   else
      OUT2_ARRAY[7] := 2;

   if (OUT1_ARRAY[8]>0.5) or (OUT1_ARRAY[9]>0.5) or
(OUT1_ARRAY[10]>0.5) or
      (OUT2_ARRAY[5]>0.5) or (OUT2_ARRAY[6]>0.5) then
            OUT2_ARRAY[7] := 1;

   if OUT2_ARRAY[7] <> 1 then
      begin (* THEN *)
         if OUT1_ARRAY[1] > 0.5 then
            OUT1_ARRAY[1] := 1
         else
            OUT1_ARRAY[1] := 0;
         if OUT2_ARRAY[1] > 0.5 then
            OUT2_ARRAY[9] := OUT2_ARRAY[9] + 1;
         if OUT2_ARRAY[2] > 0.5 then
            OUT2_ARRAY[9] := OUT2_ARRAY[9] - 1;
      end;  (* THEN *)

   rewrite(DATA_FILE);
   writeln(DATA_FILE,OUT2_ARRAY[7]);
   if OUT2_ARRAY[7] <> 2 then
      begin (* THEN *)
         if OUT2_ARRAY[10] = 1 then
```

```
writeln(DATA_FILE,TEXT1_ARRAY[trunc(OUT2_ARRAY[8])])
         else

writeln(DATA_FILE,TEXT2_ARRAY[trunc(OUT2_ARRAY[8])]);
      end; (* THEN *)
   if OUT2_ARRAY[7] <> 1 then
      begin (* THEN *)
         writeln(DATA_FILE, OUT1_ARRAY[1]);
         writeln(DATA_FILE, OUT2_ARRAY[9]);
      end;  (* THEN *)
  close(DATA_FILE);
end.  (* ANALYZE *)
```

APPENDIX D

NEURAL NETWORK TRAINING DATA

NET1 (4x6x9) TRAINING VECTORS

0.6 0.001 0.001 0.001
0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.999

0.001 0.001 0.999 0.001
0.999 0.999 0.001 0.001 0.001 0.001 0.001 0.001 0.001

0.4 0.001 0.001 0.999
0.999 0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001

0.6 0.001 0.001 0.999
0.001 0.001 0.001 0.001 0.999 0.001 0.001 0.001 0.001

0.001 0.001 0.999 0.999
0.999 0.001 0.001 0.001 0.001 0.999 0.001 0.001 0.001

0.6 0.999 0.001 0.001
0.001 0.001 0.001 0.001 0.001 0.001 0.999 0.001 0.001

0.999 0.001 0.999 0.999
0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001

0.001 0.999 0.999 0.001
0.999 0.001 0.001 0.999 0.001 0.001 0.001 0.001 0.001

0.6 0.999 0.001 0.999
0.001 0.001 0.999 0.001 0.001 0.001 0.001 0.001 0.001

0.001 0.999 0.999 0.999
0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.999 0.001

NET2 (4x4x4) TRAINING VECTORS

0.999 0.999 0.999 0.001
0.001 0.999 0.999 0.001

0.999 0.555 0.999 0.001
0.001 0.999 0.999 0.001

0.001 0.333 0.999 0.999
0.999 0.001 0.001 0.001

0.001 0.777 0.999 0.999
0.999 0.001 0.001 0.001

0.001 0.111 0.999 0.999
0.999 0.001 0.001 0.001

0.5 0.999 0.999 0.5
0.001 0.001 0.001 0.001

0.5 0.333 0.999 0.5
0.001 0.001 0.001 0.001

0.999 0.111 0.001 0.001
0.001 0.001 0.001 0.999

0.999 0.999 0.001 0.001
0.001 0.999 0.001 0.001

0.999 0.555 0.001 0.001
0.001 0.999 0.001 0.001

0.999 0.333 0.001 0.001
0.001 0.999 0.001 0.001

0.001 0.999 0.999 0.999
0.001 0.001 0.001 0.999

0.5 0.777 0.5 0.5
0.001 0.001 0.001 0.001

0.5 0.555 0.5 0.5
0.001 0.001 0.001 0.001

0.001 0.777 0.5 0.999
0.999 0.001 0.999 0.001

0.001 0.333 0.5 0.999
0.999 0.001 0.999 0.001

0.5 0.111 0.001 0.5
0.001 0.001 0.001 0.001

0.999 0.333 0.5 0.001
0.001 0.999 0.999 0.001

0.001 0.111 0.001 0.999
0.999 0.001 0.999 0.001

APPENDIX E

NEURAL NETWORK TRAINING PROGRAM SOURCE CODE

```
program TRAINER;


(*************************************************************)
(*                          TRAINER                          *)
(*************************************************************)
(*  By: Doulas M. Rogers                                     *)
(*  Last update: 8 Feb 94                                    *)
(*                                                           *)
(*   This program utilizes a gradient descent method         *)
(* to calculate the weight matrix for a neural network       *)
(* based on a backpropagation architecture.  This program    *)
(* is intended to be used to train/retrain the various       *)
(* "mini-nets" associated with the simulator control         *)
(* program (ANALYZE).  The user simply enters the number     *)
(* of input neurons ( NUM_IN ), hidden layer neurons,        *)
(* (NUM_HID), output layer neurons,(NUM_OUT), and            *)
(* filename of the training facts text file (INFILE).        *)
(* The program trains the network to a set error             *)
(* tolerance (ERROR) and outputs the weight matrix to a      *)
(* text file (OUTFILE).  Some architecture specifics:        *)
(*                                                           *)
(* Max training patterns:   const MAX_TRAINING_PAIRS (45)    *)
(* Max neurons per layer: const MAX_NEURONS (15)             *)
(* Learning rate:         const LEARN_RATE (0.2)             *)
(* Initial weights random between -0.5 and 0.5               *)
(*                                                           *)
(*************************************************************)


const
   ERROR=           0.498;(* learning tolerance parameter  *)
   LEARN_RATE=        0.2;(* learning rate parameter       *)
   MAX_NEURONS=        15;(* max # of neurons per layer    *)
   MAX_TRAINING_PAIRS=45;(* max # of input/output vectors *)


type
   ARRAYTYPE = array[1..MAX_NEURONS,1..MAX_NEURONS] of real;
   ARRAYTYPE2= array[1..MAX_NEURONS] of real;
   ARRAYTYPE3= array[1..MAX_NEURONS,1..MAX_TRAINING_PAIRS]
               of real;
   ARRAYTYPE4= array[1..MAX_TRAINING_PAIRS] of real;


var
   INFILE,                      (* training data text file  *)
   OUTFILE       : text;        (* weights export file      *)
```

```
   NUM_PATTERNS,                  (* # of training patterns   *)
   DO_PATTERN,                    (* current test pattern     *)
   LOOP1,                         (* loop control variable 1  *)
   LOOP2,                         (* loop control variable 2  *)
   NUM_IN,                        (* # input layer neurons    *)
   NUM_HID,                       (* # hidden layer neurons   *)
   NUM_OUT        : integer;      (* # output layer neurons   *)
   IN_LAYER,                      (* array input layer values *)
   DESIRED_VECTOR :ARRAYTYPE3;(* desired output patterns  *)
   HID_LAYER,                     (* hidden layer activations *)
   HID_OUTPUT,                    (* weights array hid to out *)
   OUT_OUTPUT,                    (* output layer summations  *)
   HID_ERROR,                     (* hidden layer errors      *)
   OUT_ERROR,                     (* output layer errors      *)
   OUT_LAYER      : ARRAYTYPE2;(* output layer activations *)
   WT_IN_HID,                     (* weight array inp to hid  *)
   WT_HID_OUT     : ARRAYTYPE;  (* weight array hid to out  *)
   ANSWER         : char;       (* user prompt response     *)
   CURRENT_ERROR : ARRAYTYPE4;(* largest network error    *)
   PASS_ERROR     : real;       (* largest error per pass   *)
   FILENAME       : STRING[15];(* weight export file name  *)
   STOP_FLAG      : boolean;    (* stop training flag       *)


procedure INITIALIZE_WEIGHTS( var WT_MATRIX   : ARRAYTYPE;
                                  OUTER       : integer;
                                  INNER       : integer );

var
   LOOP1,              (* outer loop control variable *)
   LOOP2  : integer; (* inner loop control variable *)

begin (* INITIALIZE_WEIGHTS *)
   for LOOP1 := 1 to OUTER do
      for LOOP2 := 1 to INNER do
         begin (* FOR *)
            WT_MATRIX[LOOP1,LOOP2] := (
               random(11)*100)+(random(10)*10) ) / 1000;
            if WT_MATRIX[LOOP1,LOOP2] > 0.5 then
               WT_MATRIX[LOOP1,LOOP2] := (WT_MATRIX[LOOP1,
                  LOOP2] - 0.5 ) * -1;
            if WT_MATRIX[LOOP1,LOOP2] < -0.5 then
               WT_MATRIX[LOOP1,LOOP2] := -0.5;
         end;  (* FOR *)
end;  (* INITIALIZE_WEIGHTS *)


procedure GET_PATTERN(     NUM_IN,
                           NUM_OUT        : integer;
```

```
                        var NUM_PATTERNS    : integer;
                        var INFILE          : text;
                        var IN_LAYER        : ARRAYTYPE3;
                        var DESIRED_VECTOR : ARRAYTYPE3 );

var
   LOOP        : integer;  (* loop control variable *)

begin (* GET_PATTERN *)
   NUM_PATTERNS := 0;
   reset(INFILE);
   while not eof(INFILE) do
      begin (* WHILE *)
         NUM_PATTERNS := NUM_PATTERNS + 1;
         for LOOP := 1 to NUM_IN do
            read(INFILE, IN_LAYER[LOOP,NUM_PATTERNS]);
         readln(INFILE);
         for LOOP := 1 to NUM_OUT do
            read(INFILE, DESIRED_VECTOR[LOOP,NUM_PATTERNS]);
      end; (* WHILE *)
   close(INFILE);
end;  (* GET_PATTERN *)



(************************ MAIN ******************************)

begin (* TRAINER *)
   clrscr;
   writeln;
   writeln('NETWORK PARAMETERS');
   writeln;

   repeat
      write('Enter number of input neurons  ---> ');
      readln(NUM_IN);
      write('Enter number of hidden neurons ---> ');
      readln(NUM_HID);
      write('Enter number of output neurons ---> ');
      readln(NUM_OUT);
   until (NUM_IN<=MAX_NEURONS) and (NUM_HID<=MAX_NEURONS)
and (NUM_OUT<=MAX_NEURONS);

   writeln;
   write('Enter training data file and extension: ');
   readln(FILENAME);
   assign(INFILE, FILENAME);

   INITIALIZE_WEIGHTS(WT_IN_HID, NUM_HID, NUM_IN);
```

```
INITIALIZE_WEIGHTS(WT_HID_OUT, NUM_OUT, NUM_HID);

GET_PATTERN(NUM_IN, NUM_OUT, NUM_PATTERNS, INFILE,
            IN_LAYER, DESIRED_VECTOR);

clrscr;
writeln;
writeln('TRAINING');
writeln;
writeln('TOLERANCE          ---> ', ERROR:7:5);
writeln('MAX CURRENT ERROR --->              ');
DO_PATTERN := 1;
STOP_FLAG := false;

repeat
   for LOOP1 := 1 to NUM_HID do
      begin (* LOOP1 *)
         HID_LAYER[LOOP1] := 0;
         for LOOP2 := 1 to NUM_IN do
            HID_LAYER[LOOP1] := HID_LAYER[LOOP1] +
               WT_IN_HID[LOOP1,LOOP2] * IN_LAYER[LOOP2,
               DO_PATTERN];
         HID_OUTPUT[LOOP1] := 1 / ( 1 + exp(
            HID_LAYER[LOOP1]));
      end;  (* LOOP1 *)

   for LOOP1 := 1 to NUM_OUT do
      begin (* LOOP1 *)
         OUT_LAYER[LOOP1] := 0;
         for LOOP2 := 1 to NUM_HID do
            OUT_LAYER[LOOP1] := OUT_LAYER[LOOP1] +
               WT_HID_OUT[LOOP1,LOOP2] *
               HID_OUTPUT[LOOP2];
         OUT_OUTPUT[LOOP1] := 1 / (1 + exp(
               OUT_LAYER[LOOP1]));
      end;  (* LOOP1 *)

   for LOOP1 := 1 to NUM_OUT do
      OUT_ERROR[LOOP1] :=
         (DESIRED_VECTOR[LOOP1,DO_PATTERN] -
         OUT_OUTPUT[LOOP1])*(exp(-OUT_LAYER[LOOP1])
         / SQR(1+exp(
         OUT_LAYER[LOOP1])));

   for LOOP1 := 1 to NUM_HID do
      begin (* LOOP1 *)
         HID_ERROR[LOOP1] := 0;
         for LOOP2 := 1 to NUM_OUT do
            HID_ERROR[LOOP1] := HID_ERROR[LOOP1] +
               OUT_ERROR[LOOP2]*WT_HID_OUT[LOOP2,LOOP1];
```

```
      HID_ERROR[LOOP1]:=HID_ERROR[LOOP1]*(exp(
         HID_LAYER[LOOP1])/SQR(1+exp(
         HID_LAYER[LOOP1])));
   end;  (* LOOP1 *)

for LOOP1 := 1 to NUM_OUT do
   for LOOP2 := 1 to NUM_HID do
      WT_HID_OUT[LOOP1,LOOP2] :=
         WT_HID_OUT[LOOP1,LOOP2] +
            LEARN_RATE*OUT_ERROR[LOOP1]*
            HID_OUTPUT[LOOP2];

for LOOP1 := 1 to NUM_HID do
   for LOOP2 := 1 to NUM_IN do
      WT_IN_HID[LOOP1,LOOP2] := WT_IN_HID[LOOP1,LOOP2]
         + LEARN_RATE*HID_ERROR[LOOP1]*
         IN_LAYER[LOOP2,DO_PATTERN];

DO_PATTERN := 0;

repeat
   DO_PATTERN := DO_PATTERN + 1;

   for LOOP1 := 1 to NUM_HID do
      begin (* LOOP1 *)
         HID_LAYER[LOOP1] := 0;
         for LOOP2 := 1 to NUM_IN do
            HID_LAYER[LOOP1] := HID_LAYER[LOOP1] +
               WT_IN_HID[LOOP1,LOOP2] *
               IN_LAYER[LOOP2, DO_PATTERN];
         HID_OUTPUT[LOOP1] := 1 / ( 1 + exp(
            HID_LAYER[LOOP1]));
      end;  (* LOOP1 *)

   for LOOP1 := 1 to NUM_OUT do
      begin (* LOOP1 *)
         OUT_LAYER[LOOP1] := 0;
         for LOOP2 := 1 to NUM_HID do
            OUT_LAYER[LOOP1] := OUT_LAYER[LOOP1] +
               WT_HID_OUT[LOOP1,LOOP2] *
               HID_OUTPUT[LOOP2];
         OUT_OUTPUT[LOOP1] := 1 / (1 + exp(
            OUT_LAYER[LOOP1]));
      end;  (* LOOP1 *)

   PASS_ERROR := 0;

   for LOOP1 := 1 to NUM_OUT do
      begin (* LOOP1 *)
         if (abs(DESIRED_VECTOR[LOOP1,DO_PATTERN]-
```

```
                OUT_OUTPUT[LOOP1])) > PASS_ERROR then
                   PASS_ERROR :=
                      abs(DESIRED_VECTOR[LOOP1,DO_PATTERN]-
                      OUT_OUTPUT[LOOP1]);
             end;  (* LOOP1 *)

          CURRENT_ERROR[DO_PATTERN] := PASS_ERROR;

       until DO_PATTERN = NUM_PATTERNS;

       DO_PATTERN := 1;

       for LOOP1 := 2 to NUM_PATTERNS do
          if (CURRENT_ERROR[LOOP1] >
          CURRENT_ERROR[DO_PATTERN]) then
             DO_PATTERN := LOOP1;
       if CURRENT_ERROR[DO_PATTERN] < ERROR then
           STOP_FLAG := true;

       gotoxy(24,5);
       write(CURRENT_ERROR[DO_PATTERN]:7:5);

   until STOP_FLAG;

   clrscr;
   writeln;
   writeln('TRAINING COMPLETE');
   writeln;
   write('View results before export? (Y/N) ---> ');
   repeat
   until keypressed;
   read(kbd,ANSWER);
   if ANSWER in ['Y','y'] then
      begin (* THEN *)
         clrscr;
         DO_PATTERN := 0;

         repeat
            DO_PATTERN := DO_PATTERN + 1;

            for LOOP1 := 1 to NUM_HID do
               begin (* LOOP1 *)
                  HID_LAYER[LOOP1] := 0;
                  for LOOP2 := 1 to NUM_IN do
                     HID_LAYER[LOOP1] := HID_LAYER[LOOP1] +
                        WT_IN_HID[LOOP1,LOOP2] *
                        IN_LAYER[LOOP2,DO_PATTERN];
                  HID_OUTPUT[LOOP1] := 1 / ( 1 + exp(
                     -HID_LAYER[LOOP1]));
               end;  (* LOOP1 *)
```

```
            for LOOP1 := 1 to NUM_OUT do
               begin (* LOOP1 *)
                  OUT_LAYER[LOOP1] := 0;
                  for LOOP2 := 1 to NUM_HID do
                     OUT_LAYER[LOOP1] := OUT_LAYER[LOOP1] +
                        WT_HID_OUT[LOOP1,LOOP2] *
                        HID_OUTPUT[LOOP2];
                  OUT_OUTPUT[LOOP1] := 1 / (1 + exp(-
                     OUT_LAYER[LOOP1]));
               end;  (* LOOP1 *)

            writeln;
            writeln('Pattern #',DO_PATTERN);
            write('IN  :');
            for LOOP1 := 1 to NUM_IN do
               write(IN_LAYER[LOOP1,DO_PATTERN]:7:4,' ');
            writeln;
            write('OUT :');
            for LOOP1 := 1 to NUM_OUT do
               write(DESIRED_VECTOR[LOOP1,DO_PATTERN]:7:4,'
                  ');
            writeln;
            textcolor(lightred);
            write('OUT :');
            for LOOP1 := 1 to NUM_OUT do
               write(OUT_OUTPUT[LOOP1]:7:4,' ');
            textcolor(yellow);
            writeln; writeln;
            write('<press ENTER to continue>');
            readln;
         until DO_PATTERN = NUM_PATTERNS;
      end;   (* THEN *)
   clrscr;
   writeln;
   writeln('EXPORT WEIGHT MATRIX');
   writeln;
   write('Enter export file name and extension ---> ');
   readln(FILENAME);

   (* EXPORT WEIGHTS FILE HERE *)

   assign(OUTFILE,FILENAME);
   rewrite(OUTFILE);
      for LOOP1 := 1 to NUM_HID do
         for LOOP2 := 1 to NUM_IN do
            writeln(OUTFILE, WT_IN_HID[LOOP1,LOOP2]);
      for LOOP1 := 1 to NUM_OUT do
         for LOOP2 := 1 to NUM_HID do
            writeln(OUTFILE, WT_HID_OUT[LOOP1,LOOP2]);
```

```
   close(OUTFILE);

   writeln;
   writeln('ALL DONE.');
end.   (* TRAINER *)
```

BIBLIOGRAPHY

[1] Juptner, Von J.H., Heat Energy and Fuels. New York: McGraw, 1908.

[2] Johnson, A.J. and G.H. Auth, Fuels and Combustion Handbook. New York: McGraw-Hill, 1951.

[3] Wenzl, Hermann F.J., The Chemical Technology of Wood. New York: Academic Press, 1970.

[4] Miller, T.W., R.S. Sutton, and P.J. Werbos, Neural Networks for Control. Cambridge, MA: MIT Press, 1990.

[5] Anderson K.L., Blankenship G.L., and Lebow, "A Rule Based Adaptive PID Controller," Proceedings of the 27th IEEE Conference on Decision and Control, I, 1 (1988), 564-569.

[6] Ignizio, J.P., Introduction to Expert Systems: the Development and Implementation of Rule-Based Expert Systems. New York: McGraw-Hill, 1991.

[7] Kosko, B., Neural Networks and Fuzzy Systems: A Dynamic Systems Approach to Machine Intelligence. Prentice-Hall, 1992.

[8] Uhr, L.M., Multi-Computer Architectures for Artificial Intelligence. New York: John Wiley & Sons, 1987.

[9] Badiru, A.B., Expert Systems Applications in Engineering and Manufacturing. Prentice-Hall, 1992.

[10] Freeman, J.A., and D.M. Skapura, Neural Networks: Algorithims, Applications, and Programming Techniques. Addison-Wesley, 1992.

[11] Antsaklis, P.J., "Neural Networks in Control Systems," IEEE Control Systems Magazine (April 1990): 3-5.

[12] Chen, F.C., "Back-Propagation Neural Networks for Nonlinear Self-Tuning Adaptive Control," IEEE Control Systems Magazine (April 1990: 44-48.

[13] Huang, S.C., and Y.F. Huang, "Learning Algorithms for Perceptrons Using Back-Propagation with Selective Updates," IEEE Control Systems Magazine (April 1990): 56-61.

[14] Liu, H., T. Iberall, and G.A. Bekey, "Neural Network Architecture for Robot Hand Control," IEEE Control Systems Magazine (April, 1989): 38-42.

[15] Passino, K.M., M.A. Sartori, and P.J. Antsaklis, "Neural Computing for Numeric-to-Symbolic Conversion in Control Systems," IEEE Control Systems Magazine (April, 1989): 44-52.

[16] Chester, D., D. Lamb, and P. Dhurjati, "Rule-Based Computer Alarm Analysis in Chemical Process Plants," Proceedings of the Seventh Annual Micro-Delcon '84, Newark, (1984), 22-29.

VITA

Douglas Michael Rogers was born on April 28, 1968 in St. Louis, Missouri. He attended public schools in St. Louis and graduated from Lindbergh High School in 1986. From 1986 to 1987 he attended Wentworth Military Academy & Junior College in Lexington, Missouri on a scholarship awarded by The Falcon Foundation. In 1987 he received an appointment to the U.S. Air Force Academy in Colorado Springs, Colorado where he received a Bachelor of Science degree in Electrical Engineering in 1991.

Upon graduation, Lieutenant Rogers attended Air Force undergraduate pilot training at Columbus Air Force Base, Mississippi where he earned his wings in 1992.

In January 1993, Lt. Rogers was sent by the Air Force Institute of Technology to the University of Missouri-Rolla where he is currently enrolled as a graduate student.